**UNIVERSIDAD AUTÓNOMA DE MADRID**
**ESCUELA POLITÉCNICA SUPERIOR**


**Grado en Ingeniería informática**

# TRABAJO FIN DE GRADO

## scikit-fda: ANOVA de un factor y test $T^2$ de Hotelling

**Autor: David García Fernández**
**Tutor: Alberto Suárez González**

**junio 2020**

**David García Fernández**
***scikit-fda: ANOVA de un factor y test $T^2$ de Hotelling***

**David García Fernández**

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

# Agradecimientos

Quiero mostrar mi agradecimiento a todos los miembros que hay detrás de `scikit-fda`. En especial a mi tutor, Alberto Suárez por sus comentarios y apuntes durante las reuniones, y a todo el tiempo que ha dedicado a este proyecto. A Carlos Ramos Carreño, por compartir su amplio conocimiento sobre `Python` a base de sugerencias y consejos, y por supuesto por sus exhaustivas revisiones del código. A José Luis Torrecilla y Luis Alberto Rodríguez les agradezco sus explicaciones y aclaraciones en los aspectos matemáticos. Además, me gustaría reconocer la labor de mi compañero Yujian Hong, que desde el comienzo de la carrera universitaria se ha mostrado siempre dispuesto a echarme una mano.

Por último les quiero agradecer todo el apoyo y la compresión a mis padres, a mi hermano Mario y a María José. Sin ellos esto no se podría haber conseguido.

# Resumen

El análisis de datos funcionales es una disciplina cuyo objeto de estudio son los datos que, por su naturaleza, pueden ser entendidos como funciones que dependen de un parámetro continuo. Se trata de una rama de la estadística que se ha desarrollado en los últimos veinte años. Los datos funcionales suponen un cambio de paradigma, ya que toman valores en espacios de dimensión infinita. Por esta razón no es trivial generalizar muchas de las técnicas que se utilizan en la estadística multivariante a este nuevo contexto.

Las principales herramientas de programación especializadas en el análisis de este tipo de datos se encuentran escritas en lenguaje `R`. La librería `scikit-fda` [9] pretende ser una alternativa *open source* para los usuarios de `Python`. Este lenguaje es uno de los que más ha crecido en los últimos años. Gracias a su sintaxis simple y de alto nivel ha llamado la atención de numerosos miembros de la comunidad científica.

En este trabajo se ampliará la funcionalidad de `scikit-fda` con dos contrastes de hipótesis estadísticos, siempre dentro del contexto funcional. El primero de ellos es un contraste ANOVA de un factor, útil para medir la influencia de un factor categórico sobre una variable cuantitativa. Por ejemplo, si se desea determinar si el tipo de abono utilizado en una población de flores ha influido en el crecimiento de sus pétalos. El segundo contraste es un test de igualdad de medias entre dos poblaciones basado en la $T^2$ de Hotelling, que puede utilizarse como complemento del anterior en un análisis *post-hoc*.

En este texto se tratan los aspectos teóricos subyacentes. En concreto se exploran los conceptos de dato funcional y contraste de hipótesis, necesarios para el posterior estudio de los contrastes de hipótesis implementados. También se detallan las diferentes fases del desarrollo de estas herramientas, dentro de la librería. Por último, se muestran varios ejemplos, con datos sintéticos y reales, del uso de estas técnicas y su integración con el resto de la funcionalidad disponible en `scikit-fda`.

# Palabras clave

Análisis de datos funcionales, ANOVA, Hotelling, contrastes de hipótesis, Python, scikit-fda

# Abstract

Functional data analysis is a field of the Statistics where each datum is a continuous function. This area has been developed over the last twenty years. In order to manage functional data a change of perspective is needed, as it is necessary to work in infinite-dimensional spaces. For this reason, it is not trivial to generalize some classical statistic techniques in $\mathbb{R}^p$ spaces.

The main functional data analysis programming tools are written in the `R` language. `scikit-fda` library [9] aims to be an open source alternative for `Python` users. This language is one of the fastest growing in recent years. Due to its simple and high-level syntax, it has attracted the attention of many members of the scientific community.

In this work, the functionality of `scikit-fda` is extended with two statistical hypothesis tests in the functional context. The first test is based on One-way ANOVA, and it is useful to determine the influence of a predictor variable over a response variable. For example, measuring the effect of the fertilizer used on a population of roses with respect to the growth of their petals. The second one determines if the means of two samples are significantly different from each other and it is based on Hotelling's $T^2$. Both tests can be used in a complementary way.

This text deals with the underlying theoretical aspects, such as the concept of functional data and hypothesis testing, which are necessary to understand both tests. The different stages in the development of this software tools are also detailed. Finally, some examples of the use and integration of these methods are presented, using synthetic and real data available in `scikit-fda`.

# Keywords

Functional data analysis, ANOVA, Hotelling, statistical hypothesis testing, scikit-fda, Python

# Índice

# Listas

## Lista de ecuaciones



## Lista de figuras

# Lista de tablas

# 1

# Introducción

El Análisis de Datos Funcionales (FDA) es una rama de la estadística cuyo objeto de estudio son funciones que dependen de un parámetro continuo. Esto supone un cambio de perspectiva respecto a la Estadística clásica, principalmente desarrollada en espacios de dimensión finita como $\mathbb{R}^p$.

Este campo comenzó a ser estudiado a mediados del s.XX. Sin embargo, se suele identificar su origen como especialidad con la aparición de la primera monografía sobre el tema, el libro *Functional Data Analysis*, de James O. Ramsay [19], en 1997. Hoy en día el uso del FDA se ha extendido en numerosos ámbitos, como la Biomedicina, la Economía y el Aprendizaje Automático. Una de las razones más claras de su rápida evolución ha sido los avances tecnológicos que se han dado en las últimas décadas, por los que es posible registrar y manejar observaciones de alta dimensionalidad de manera sencilla.

Por otro lado, el *diseño de experimentos* es un conjunto de técnicas estadísticas muy utilizadas por la comunidad científica. Uno de sus principales objetivos es cuantificar la influencia de varios factores categóricos (por ejemplo, el día de la semana y el mes del año) sobre una variable respuesta (en el mismo ejemplo, el número de espectadores en una sala de cine). La más sencilla de estas técnicas, el Análisis de la varianza (ANOVA) de un factor, es la que trataremos en este trabajo. Este método permite estudiar el efecto de un único agente, a través de una descomposición de la variabilidad de los datos de una muestra.

Este proyecto se centra en la implementación de la técnica ANOVA de un factor en el contexto funcional, dentro de la librería `scikit-fda`. El proyecto `scikit-fda` comenzó en 2018, como parte de un trabajo de fin de grado [1]. A día de hoy ha crecido y está dirigido por miembros del Grupo de Aprendizaje Automático de la Escuela Politécnica Superior, en la Universidad Autónoma de Madrid. Su propósito es implementar herramientas de FDA , para hacerlas accesibles a los usuarios del lenguaje `Python`. Ya existen librerías en `R` orientadas al manejo de datos funcionales, como `fda`, promovida por Ramsay [17] y `fda.usc` desarrollada en la Universidad de Santiago de Compostela [5]. `R` es un lenguaje muy utilizado por los profesionales e investigadores en Estadística, sin embargo en los últimos años `Python` ha despertado el interés de numerosos miembros de la comunidad científica. Existen múltiples librerías `Python` especializadas en el área del análisis de datos, como pueden ser `numpy` o

`scipy`. Ésta última contiene una serie de procedimientos matemáticos que han servido de base para librerías más especializadas, los llamados *scikit*. Entre ellos el más reconocido es `scikit-learn`, para Aprendizaje Automático, sin embargo existen a día de hoy más de cincuenta [1], enfocados en campos heterogéneos como la química o el procesamiento de imágenes. La librería `scikit-fda` pertenece a este conjunto de paquetes especializados, y se mantiene integrada con los principales módulos de uso común en `Python`. Además, se trata de un proyecto *open source*, que promueve la colaboración abierta de cualquier usuario.

## 1.1. Motivación

La principal motivación de este trabajo es la de ampliar la funcionalidad de `scikit-fda` con metodología útil en el diseño de experimentos estadísticos. Por un lado se planteará un contraste de hipótesis basado en el modelo ANOVA . Por otro lado, se incluirá un contraste de igualdad de medias entre dos muestras, el contraste $T^2$ de Hotelling. Como se verá en el texto, ambos contrastes se pueden utilizar de manera combinada. Al tratarse de técnicas de uso común se consideran una herramienta básica con la que la librería debe contar.

## 1.2. Objetivos

Exploraremos el concepto de dato funcional, revisando sus propiedades y estudiando cómo se redefinen algunas nociones estadísticas básicas bajo este nuevo contexto. Por otro lado, tras describir brevemente la teoría que hay tras el ANOVA de un factor, se pretende revisar una de las alternativas propuestas hasta la fecha en el contexto funcional, la cual será analizada en profundidad para su posterior implementación en la librería `scikit-fda`. El mismo procedimiento se seguirá para un contraste de igualdad de medias basado en el estadístico $T^2$ de Hotelling. Además, se realizarán varios estudios de simulación que ilustren el funcionamiento de estos procedimientos en la librería, a modo de ejemplos de uso, así como experimentos para medir la eficiencia de las implementaciones en términos de precisión.

## 1.3. Estructura del documento

En el Capítulo 2 se mostrarán los conceptos sobre datos funcionales (fundamentos matemáticos y diferentes formas de representación). Se continuará definiendo las propiedades básicas de los contrastes de hipótesis, el contraste ANOVA clásico y el análisis a posteriori. La pieza central del capítulo

[1] https://www.scipy.org/scikits.html

será el análisis de dos propuestas bastante recientes para la aplicación de los contrastes citados.

En el Capítulo 3 se explica el proceso de desarrollo de la nueva funcionalidad. Se realizará una pequeña exploración de los requisitos de la librería, de las decisiones de diseño escogidas y de la metodología de desarrollo seguida.

Por último, en el Capítulo 4 se realiza un estudio de simulación, a modo de ilustración de las propiedades y la utilidad del software implementado, sobre datos sintéticos y reales. El trabajo se cierra con las conclusiones, en el Capítulo 5.

# 2

# Estado del arte

En este capítulo vamos a presentar el concepto de dato funcional, revisando sus propiedades y las herramientas necesarias para su manipulación. Por otro lado, se estudiarán dos propuestas de actualidad enfocadas en la implementación de los contrastes ANOVA y de la $T^2$ de Hotelling para igualdad de medias.

## 2.1. Datos funcionales

Los datos funcionales son, en esencia, realizaciones de variables aleatorias que toman valores en espacios de dimensión infinita [6, Definición 1.1]. Sin embargo, si queremos concretar un poco más esta definición podemos remontarnos al primer artículo que menciona de manera explícita el concepto de dato funcional, *When the data are functions*, de J. O. Ramsay [18]. En él se trata a este tipo de datos como funciones continuas, dependientes de un parámetro $t$ sobre un conjunto cerrado $T$.

En el artículo se explica que lo más característico de los datos funcionales, que les distingue de los datos multivariantes, es que los valores $x(t_1)$ y $x(t_2)$ del dato serán muy similares si $t_1$ y $t_2$ están cerca el uno del otro.

No obstante, no existe en la literatura una definición precisa y consensuada de lo que es un dato funcional. En este trabajo los datos funcionales se consideran funciones continuas $x(t)$, tomando valores en un intervalo cerrado $[a, b]$ de la recta real, con cierto grado de suavidad (esto se traduce en la existencia de las primeras derivadas del dato). Esta definición ha sido construida tomando distintos conceptos que se tratan en las publicaciones [10], [11], [12], y en las anteriormente citadas. A continuación se tratarán algunos conceptos de especial relevancia en el tratamiento de datos funcionales.

### 2.1.1. Fundamentos matemáticos

Una variable aleatoria funcional puede entenderse como un proceso estocástico continuo.

Un proceso estocástico $X(t)$ se define como una colección ordenada de variables aleatorias, inde-

xada por un parámetro $t \in T$:

$$X(t) := \{X_t : \Omega_t \to S_t,\ t \in T\}.$$

A lo largo del trabajo llamaremos simplemente proceso a una variable aleatoria funcional. Como podemos ver, un proceso es la generalización de un vector aleatorio a la dimensión infinita. Por esta razón algunos objetos como la matriz de covarianza también se generalizan.

La función de covarianza $K$ de un proceso $X(t)$ es aquella que se define como

$$K(s,t) = Cov(X_s, X_t).$$

Un tipo especial de procesos, los procesos gaussianos, serán muy utilizados durante este texto. Dado un proceso estocástico $Z(t)$, diremos que $Z(t)$ es un proceso gaussiano si $(Z(t_1), \ldots, Z(t_M))^T$ es una normal multivariante, para cualquier subconjunto finito de índices $\{t_i\}_{i=1}^M$, es decir, un vector aleatorio para el que cualquier combinación lineal de sus componentes

$$a_1 Z_1 + \cdots + a_M Z_M$$

está normalmente distribuida.

Los procesos que hemos tratado en esta sección toman valores en espacios de dimensión infinita. En nuestro caso trabajaremos sobre espacios de Hilbert, que se caracterizan por tener un producto interno. El espacio de Hilbert que más utilizaremos es $L^2([a,b])$, el espacio de funciones con cuadrado integrable en el intervalo de la recta real $[a,b]$. El producto interno en $L^2([a,b])$ entre dos elementos $f$ y $g$ se define como:

$$\langle f, g \rangle = \int_a^b f(x)g(x)dx.$$

La norma de un elemento $f$ inducida por este producto interno se denota como $\|f\| = \sqrt{\langle f, f \rangle}$.

Este producto interno nos permite definir el concepto de operador de covarianza. El operador de covarianza en estadística multivariante en dimensión finita se representa a través de una matriz, al igual que la función de covarianza. Sin embargo, esto no es posible en el caso funcional. Siendo $P$ la medida de Lebesgue, el operador de covarianza se puede definir como aquel $\mathcal{K} : L^2([a,b]) \to L^2([a,b])$ que cumple:

$$\langle \mathcal{K}x, y \rangle = Cov(x,y) = \int_a^b \langle x, z \rangle \langle y, z \rangle dP(z), \tag{2.1}$$

para $x, y \in L^2([a,b])$.

Para más información sobre espacios de Hilbert se puede consultar el libro de Conway [2].

### 2.1.2. Representación

Por mucha precisión que podamos conseguir de un instrumento de medida, el conjunto de valores que podremos registrar de un dato funcional es finito. De esta manera, una muestra de un dato funcional habrá sido recogida en un conjunto finito, de $M$ puntos $t_1 < \ldots < t_M$. La observación del dato se indica entonces como un conjunto de pares $\{(t_i, x_i)\}_{i=1}^{M}$.

En consecuencia, una primera opción es tratar los datos como vectores, igual que en la estadística multivariante. Sin embargo, perderíamos las características esenciales de los datos funcionales. La primera de ellas es la continuidad: es deseable poder conocer el valor $x(t)$, para un $t$ cualquiera en el intervalo $[a, b]$.

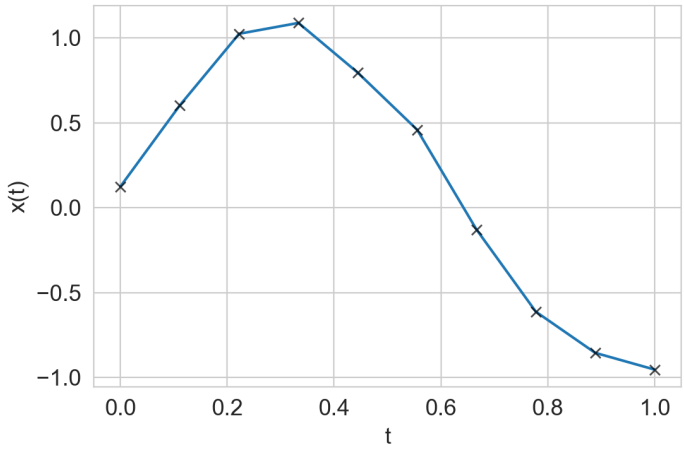



**Figura 2.1:** Representación discreta de una muestra de un dato funcional. En aspas negras se presentan los 10 valores de la muestra, tomados en puntos equiespaciados del intervalo $[0, 1]$. En azul podemos ver una interpolación lineal local de los mismos.

En la Figura 2.1 se ha utilizado una interpolación lineal como primera aproximación a este problema, es decir, los puntos se han unido mediante segmentos rectilíneos. A pesar de recuperar la continuidad que se había perdido al discretizar la muestra es evidente que este ajuste no parece el más adecuado para los datos. Además, es posible que se cometan errores al tomar las muestras, que deseamos *suavizar*, es decir, no queremos que nuestra representación del dato se ciña total y necesariamente a los valores observados.

La *representación en bases* [19, Sección 3.3] solventa estos problemas. Éste método consiste en escoger una base de un subespacio de dimensión finita de nuestro espacio de funciones. Esta base es similar a la de un espacio vectorial, donde cada elemento es una función continua, normalmente diferenciable, e independiente del resto. De esta manera podemos representar un dato funcional como una combinación lineal de los elementos de la base. Si suponemos una base con $k$ elementos escribiremos nuestro dato funcional como:

$$x(t) = \sum_{i=1}^{k} c_i \phi_i(t) \tag{2.2}$$

Donde $(c_1, \ldots, c_k)$ es una colección de coeficientes y $(\phi_1, \ldots, \phi_k)$ las funciones de la base.

Gracias a la representación en bases podemos, además, estudiar las derivadas de los datos, característica de gran relevancia en el análisis de datos funcionales.

Los factores que influyen a la hora de realizar una representación por bases son, por un lado, el número $k$ de elementos, y por otro las funciones $\phi_i, i = 1, \ldots, k$ que escojamos. Es importante elegir estos parámetros de modo que las características que conocemos a priori sobre los datos se vean reflejadas en la representación y no queden distorsionadas.

Por ejemplo, un número $k = M$ de elementos nos puede proporcionar una interpolación de los datos observados. Por el posible error cometido al tomar la muestra que se ha comentado anteriormente puede ser interesante escoger valores de $k < M$. En general, valores pequeños de $k$ aligeran la computación, y aumentan los grados de libertad en el caso de querer probar hipótesis sobre los datos.

Por otro lado, es importante escoger una base que comparta características similares a las que tienen nuestros datos.

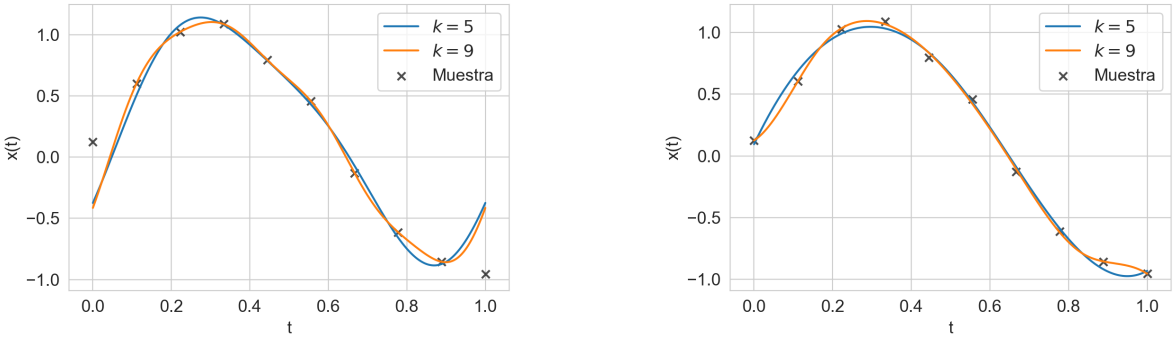
Un ejemplo simple de base de funciones es la base de monomios, donde se define cada elemento como $\phi_i(t) = (t - \omega)^i$. Sin embargo las bases que más utilizaremos en este trabajo son las siguientes. Empezaremos con la de Fourier, definida como:

$$\phi_0(t) = 1, \quad \phi_{2r-1}(t) = \sin(r\omega t), \quad \phi_{2r} = \cos(r\omega t). \tag{2.3}$$

Se caracteriza por ser periódica si su dimensión es finita, con periodo $2\pi/\omega$, y por que es posible calcular los coeficientes $c_1, \ldots, c_k$ de manera sencilla a través de la transformada rápida de Fourier. Esta base suele ser la indicada para funciones sin características muy prominentes, con una curvatura similar en toda su imagen.

Para datos no periódicos, la aproximación preferida suele ser la representación en bases de “splines”. Las splines son bases fáciles de computar, por guardar parecido a los polinomios, a la par que permiten una mayor flexibilidad de representación. No entraremos en detalles técnicos sobre la definición de este tipo de bases.

Relacionado con el concepto de base tenemos las matrices de Gram. Dada una base de funciones $\{\phi_i\}_{i=1}^k$ se define la matriz de Gram como aquella matriz $\mathbf{W}$ tal que $\mathbf{W}_{jk} = \langle \phi_j, \phi_k \rangle$.




(a) Representación en bases de Fourier

(b) Representación en bases de B-Splines

**Figura 2.2:** En la imagen se ilustra la representación a través de bases de distinto tipo y dimensión de una misma muestra de un único dato funcional. Se aprecia que a mayor número de bases el ajuste a los valores observados es mayor. También se aprecia que la representación en splines es más flexible que la de Fourier, que muestra una marcada naturaleza periódica.

## 2.2. Contrastes de hipótesis

El objetivo principal de este TFG es la implementación de dos contrastes de hipótesis. Es necesario, por tanto, una revisión de los conceptos clave que debemos considerar a la hora de afrontar un problema de este tipo. En esencia, un contraste consistirá en tomar decisiones acerca de la población que se estudia en base a la muestra de datos que disponemos.

Los contrastes estadísticos se construyen sobre hipótesis nulas. Una hipótesis nula es una suposición que hacemos sobre la población, normalmente sobre su distribución de probabilidad. Se denotan como $H_0$, mientras que a las hipótesis alternativas (las contrarias a $H_0$), se las denota por $H_1$.

Para ejecutar el test, prueba, o contraste de hipótesis comprobamos si los valores observados encajan con los resultados esperados bajo la hipótesis nula. En caso de que la diferencia entre ellos sea grande diremos que hay suficiente evidencia estadística para rechazar $H_0$.

Existen dos tipos de errores relacionados con este hecho. El primero es el *error de tipo I*, que ocurre si rechazamos erróneamente una hipótesis nula. Por otro lado, el *error de tipo II*, consiste en aceptar la hipótesis nula cuando esta es falsa. Llamaremos *nivel de significación* $\alpha$ a la probabilidad de cometer un error de tipo I que estamos dispuestos a asumir como máximo al tomar una decisión sobre $H_0$. En la práctica se suele escoger $\alpha = 5\,\%$, pero es un parámetro a considerar según las características del problema.

Algunos contrastes, como los que se implementarán aquí, se basan en el valor de un estadístico. Un estadístico $S$ es una variable que depende de los parámetros de la población estudiada. Supongamos que $S_n$ es el valor empírico del estadístico, es decir el que se estima a través de la muestra de datos.

Rechazaremos $H_0$ con un nivel de significación $\alpha$ en caso de que $S_n > S_\alpha$, donde $S_\alpha$ es el valor tal que:

$$\mathrm{P}(S > S_\alpha | H_0) = \alpha. \tag{2.4}$$

Por otro lado tenemos el concepto de p-valor, que representa la probabilidad de obtener un valor de $S$ más extremo que $S_n$. Se define como:

$$p_{valor} = \mathrm{P}(S > S_n | H_0). \tag{2.5}$$

Por tanto, si queremos rechazar $H_0$ con un nivel de significación $\alpha$ el p-valor obtenido debe ser más pequeño que $\alpha$.

Para más detalles acerca de las pruebas de hipótesis se puede consultar, por ejemplo, el libro *Probabilidad y Estadística* de Spiegel y Schiller [20, Capítulo 7].

En resumen, los pasos a seguir al realizar un contraste de hipótesis basado en un estadístico $S$ son:

1.– Inferir o aproximar la distribución de $S$ bajo la hipótesis nula.

2.– Calcular el valor empírico $S_n$, a partir de la muestra.

3.– Calcular el p-valor a partir de los resultados de los pasos anteriores, utilizando la Ecuación (2.5).

Un algoritmo basado en estos tres pasos nos proporciona el valor $S_n$ y el p-valor.

El punto crítico suele ser el primero: el cálculo de la función de distribución de $S$ bajo $H_0$. En algunos casos se puede conocer de manera analítica el valor de la distribución de antemano. En otros, como ocurre en los contrastes que trataremos, esto no es posible.

## 2.3. Diseño de experimentos

En esta sección resumiremos los conceptos en los que se basa el diseño de experimentos y en concreto la técnica ANOVA de un factor. Como se comentó en el Capítulo 1 el principal uso de esta técnica es determinar si un factor categórico o discreto, que actúa en $I$ niveles diferentes, influye de manera significativa sobre el valor de una variable respuesta $X(t)$.

A modo de ejemplo utilizaremos un subconjunto del *dataset* de Canadian Weather [19, Sección 1.3], que se puede ver representado en la Figura 2.3. Este conjunto de datos contiene información sobre la temperatura media anual registrada en distintas estaciones meteorológicas de Canadá entre 1960 y 1994. Los datos se encuentran etiquetados por tipo el tipo de clima de la localización de la estación.

Vamos a considerar como factor el tipo de clima, con $I = 3$ y la temperatura anual $X(t)$ como la

variable respuesta. Nuestro objetivo es determinar si el tipo de clima en el que hemos recogido las observaciones influye en la temperatura que se ha registrado.

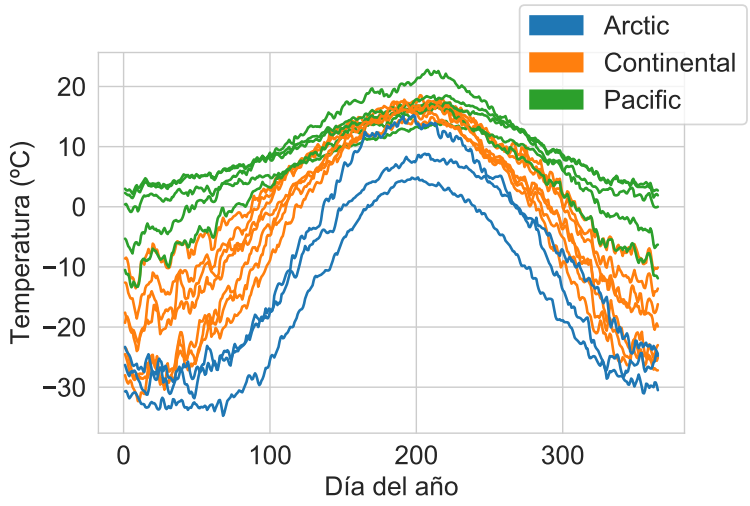



**Figura 2.3:** Trayectorias seleccionadas y etiquetadas según el clima. En azul se representan las temperaturas registradas en clima ártico, en naranja en clima continental y en verde en clima pacífico. Los datos han sido tomados del dataset Canadian Weather.

Para abordar el problema se propone el siguiente modelo lineal sobre los datos. Supongamos que disponemos de una muestra de tamaño $n$ de un conjunto de procesos $\{X_{ij}(t)\}$ independientes, donde $i = 1, \ldots, I$, indica el nivel del factor al que corresponde cada proceso y $j = 1, \ldots, n_i$, donde $n_i$ representa el número de procesos pertenecientes al i-ésimo nivel del factor.

En la muestra de la Figura 2.3, tenemos tres observaciones que corresponden al clima ártico ($n_1 = 3$), siete al continental ($n_2 = 7$) y cinco al pacífico ($n_3 = 5$), haciendo un total de $n = n_1 + n_2 + n_3 = 15$ datos.

Con estos elementos se define el modelo lineal:

$$X_{ij}(t) = m_i(t) + e_{ij}(t), \quad i = 1, \ldots, I, j = 1, \ldots, n_i, \tag{2.6}$$

donde $m_i(t)$ es la media en el nivel i-ésimo ($\mathrm{E}(X_{ij}(t)) = m_i(t)$, para un mismo $i$), que se relaciona con el efecto del factor sobre el dato. El residuo, $e_{ij}(t)$, representa efectos aleatorios adicionales, que no provienen del factor que estudiamos.

Como hipóteis sobre el modelo se supone que los residuos se distribuyen como procesos gaussianos independientes, centrados (media cero), y con misma función de covarianza $K(s,t)$ (homocedasticidad). Por esta razón:

$$X_{ij}(t) \sim \mathcal{G}(m_i(t), K(s,t)), \text{ independientes.} \tag{2.7}$$

### Estimación de parámetros

La media de cada nivel del factor $m_i(t)$ se estimará mediante el promedio de los datos de cada nivel:

$$\hat{m}_i(t) = \bar{X}_i(t) = \frac{1}{n_i}\sum_{j=1}^{n_i} X_{ij}(t). \tag{2.8}$$

Estimaremos la función de covarianza como:

$$\hat{K}(s,t) = \frac{1}{n-1}\sum_{i=1}^{I}\sum_{j=1}^{n_i}\left(X_{ij}(s) - \bar{X}(s)\right)\left(X_{ij}(t) - \bar{X}(t)\right). \tag{2.9}$$

El operador de covarianza en $L^2([a,b])$ queda determinado por la función de covarianza $K$, como vimos en la Ecuación (2.1). Equivalentemente podemos utilizar el siguiente estimador, que utilizaremos en el caso homocedástico:

$$\hat{\mathcal{K}} = \mathcal{K}_n = \frac{1}{n-1}\sum_{i=1}^{I}\sum_{j=1}^{n_i}(X_{ij}(t) - \bar{X}(t)) \otimes (X_{ij}(t) - \bar{X}(t)), \tag{2.10}$$

donde $(f \otimes g) : \mathbb{H} \to \mathbb{H} : h \mapsto \langle f, h\rangle g$ denota el producto tensorial [11, Definición 3.4.6].

En las expresiones anteriores $\bar{X}$ representa el promedio de todos los datos de la muestra, que se define como

$$\hat{m}(t) = \bar{X}(t) = \frac{1}{n}\sum_{i=1}^{I} n_i \bar{X}_i(t).$$

Como se ha puntualizado más arriba, los residuos $e_{ij}(t)$ son valores aleatorios que no guardan relación con el factor que estudiamos. Sin embargo en $m_i(t)$ es donde se recoge toda la información característica del nivel del factor. Este hecho nos permite diseñar una hipótesis nula

$$H_0 : m_1(t) = m_2(t) = \cdots = m_I(t). \tag{2.11}$$

En caso de que $H_0$ sea cierta no habrá diferencias entre las medias de cada nivel, esto implica que el efecto del factor es nulo, pues no genera discrepancias entre los distintos niveles. En caso contrario, al rechazarla, sabremos que al menos una de las medias difiere del resto, por lo que el factor ha influido sobre la población.

La técnica ANOVA nos permitrá definir un estadístico con el que poder ejecutar el contraste sobre la hipótesis nula.

La idea en la que se basa el análisis de la varianza es en la descomposición de la variabilidad total de los datos entre la variabilidad explicada por los $m_i(t)$ del modelo y la residual de los $e_{ij}(t)$.

En el enfoque clásico (en $\mathbb{R}$) esta descomposición se lleva a cabo de la siguiente manera:

$$SCT = SCE + SCR. \tag{2.12}$$

Donde cada sumando representa:

- **Suma de cuadrados total:** Se trata de la variabilidad total de la muestra.

$$SCT = \sum_{i=1}^{I}\sum_{j=1}^{n_i}(X_{ij} - \bar{X})^2. \tag{2.13}$$

- **Suma de cuadrados explicada:** Variabilidad que explica el modelo.

$$SCE = \sum_{i=1}^{I} n_i(\bar{X}_i - \bar{X})^2. \tag{2.14}$$

- **Suma de cuadrados residual:** Suma de los efectos de los residuos.

$$SCR = \sum_{i=1}^{I}\sum_{j=1}^{n_i}(X_{ij} - \bar{X}_i)^2 = \sum_{i=1}^{I}\sum_{j=1}^{n_i} e_{ij}^2. \tag{2.15}$$

Siendo $\bar{X}_i$ el promedio para el i-ésimo nivel del factor.

El estadístico del contraste clásico se define bajo estos términos:

$$F = \frac{SCE/(I-1)}{SCR/(n-I)}. \tag{2.16}$$

Su nombre $F$ es en honor al matemático y genetista Ronald Fisher, que ideó esta y otras técnicas del diseño de experimentos [7]. Bajo $H_0$, por las razones que hemos comentado más arriba, se espera un valor pequeño del estadístico (poca variabilidad explicada por el modelo frente a mucha variabilidad residual). Sin embargo bajo $H_1$ ocurrirá el efecto contrario.

A la hora de implementar este contraste nos remontamos a lo que fue comentado en la Sección 2.2. Por un lado, tenemos una forma explícita de calcular el valor empírico del estadístico ($F_n$), según la Ecuación (2.16). Por otro lado, necesitamos conocer la distribución de $F$, para poder calcular el p-valor. En este caso (en $\mathbb{R}$) es sencillo obtenerla, pues según se ha definido $SCE$ y $SCR$ el estadístico se distribuye como una $F$ de Fisher:

$$F \sim F_{I-1;n-I}.$$

El p-valor se calcula en términos de dicha distribución, que es conocida y de uso común. Existen múltiples herramientas que implementan el ANOVA clásico basándose en este estadístico. La que nos servirá de referencia es `f_oneway` de la librería `scipy`. Esta función facilita el valor $F_n$ y el p-valor

del contraste [1].

### 2.3.1. ANOVA de un factor funcional

Volviendo al caso funcional nos encontramos con un problema. Si definimos un análogo del estadístico $F$ no podemos deducir de manera sencilla su distribución. En consecuencia, es necesario plantear un estadístico diferente, que permita contrastar la misma hipótesis nula (2.11) y cuya distribución podamos inferir o, al menos, aproximar.

Estudiaremos la propuesta que Cuevas, Febrero y Fraiman publicaron en el año 2004 en el artículo *An ANOVA test for functional data* [3].

Los autores plantean el uso de la siguiente variable, bajo el mismo modelo lineal unifactorial:

$$V_n = \sum_{i<j} n_i \|\bar{X}_i(t) - \bar{X}_j(t)\|^2. \tag{2.17}$$

Como vemos, es una manera alternativa de medir la variabilidad entre los distintos niveles del factor (explicada por el modelo), tal y como lo hacía $SCE$ (2.14). De nuevo, se esperan valores de $V_n$ cercanos a cero bajo $H_0$. El valor empírico de $V_n$ es sencillo de calcular, sin embargo también queremos saber cómo se distribuye. En el artículo se expone el siguiente teorema, que nos permitirá calcular una buena aproximación de la distribución de $V_n$.

**Teorema 2.1.** *Sean $n_i$ y $n$ tales que $\frac{n_i}{n} \to p_i > 0$, para $i = 1, \ldots, I$. Sean $X_{ij}(t), j = 1 \ldots, n_i$ observaciones de $I$ procesos independientes sobre $L^2([a,b])$, con media cero y funciones de covarianza $K_i(s,t)$. Consideremos también la variable:*

$$V := \sum_{i<j}^{I} \|Z_i(t) - C_{ij} Z_j(t)\|^2, \tag{2.18}$$

*donde los procesos $Z_i(t), i = 1, \ldots, I$ son procesos gaussianos independientes con media 0 y función de covarianza $K_i(s,t)$, y $C_{ij} = \sqrt{\frac{p_i}{p_j}}$.*

*Entonces, bajo la hipótesis nula, se satisface:*

$$V_n \xrightarrow{d} V$$

Este resultado nos indica que, asintóticamente (cuando el número de datos $n \to \infty$), la distribución de $V_n$ coincide con la de $V$. La ventaja que tiene utilizar $V$ frente a $V_n$ es que sus componentes son procesos gaussianos, cuyas propiedades facilitan su simulación (ver Sección 2.1.1).

Si entendemos $V$ como una variable aleatoria, podemos generar sintéticamente una muestra de

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.f_oneway.html

tamaño $B$ de la misma. Con ella seremos capaces de construir una distribución muestral o empírica de $V$, a través de la cual podemos calcular una aproximación del p-valor. Esta aproximación será precisa si utilizamos valores de $B$ lo suficientemente grandes.

En el Algoritmo 2.4 se ilustran los pasos necesarios para calcular la distribución empírica de $V$. Partimos de tantas muestras de datos `fd_i`, $i = 1, \ldots, I$, como niveles del factor.

```
Input: Datos por nivel - fd = [muestra1, ..., muestraI]
Output: Distribución empírica - dist

# Estimación de parámetros
for i ← 0 to I do:
      covarianzas[i] ← EstimarCovarianza(fd[i])
end

# Cálculo de la estimación empírica
for i ← 0 to B do
      for j ← 0 to I do
            procesos_z[j] ← SimularGaussiano(0, covarianzas[j], M)
      end
      dist[i] ← EstadisticoV(procesos_z)
end
devolver(dist)
```

**Figura 2.4:** Algoritmo para obtener una colección de muestras Monte Carlo del estadístico $V$. Existen dos pasos diferenciados, primero la estimación de parámetros y después la simulación de procesos gaussianos, que son utilizados para calcular cada observación sintética de $V$. En este ejemplo se trata el caso heterocedástico, más general.

Veamos en detalle cada paso en este algoritmo.

### Estimación de parámetros

Necesitamos estimar la matriz de covarianza de cada proceso $Z_i(t)$ a partir de los datos de la muestra. Recordemos que, bajo homocedasticidad, todos los $X_{ij}$ comparten una misma función de covarianza $K$. Esto era esencial en el método ANOVA clásico, sin embargo gracias al Teorema 2.1 podremos considerar el caso heterocedástico, en el que existe una función de covarianza $K_i(s,t)$ diferente para cada nivel del factor. De esta manera estimamos el $K_i(s,t)$, correspondiente a $Z_i(t)$, como

$$\hat{K}_i(s,t) = \frac{1}{n_i - 1} \sum_{j=1}^{n_i} \left(X_{ij}(s) - \bar{X}(s)\right)\left(X_{ij}(t) - \bar{X}(t)\right). \tag{2.19}$$

Si consideramos el caso homocedástico la estimación de la $K$ general se realizará a través de la Ecuación (2.9), con todos los datos de la muestra.

### Simulación

En este paso obtendremos un conjunto $\{\tilde{V}_b\}_{b=1}^B$ de valores simulados del estadístico $V$. Para cada uno de estos $B$ valores se requiere simular una trayectoria de cada uno de los procesos gaussianos $Z_i(t)$.

Para realizar esta simulación debemos fijar un mallado de $M$ puntos sobre el intervalo $[a, b]$ en el que toman valores los procesos. Sobre este mallado se evaluan las trayectorias simuladas. Existe, en la librería `scikit-fda`, una función que genera este tipo de trayectorias discretizadas. Por definición de proceso gaussiano sabemos que cualquier colección finita de las variables aleatorias que lo componen seguirá una distribución normal multivariante. De este modo lo que estamos simulando realmente son observaciones de una normal en $\mathbb{R}^M$.

De esta manera, para cada $b = 1, \ldots, B$, generamos un vector normal

$$Z_{ib}^* = (Z_{ib}^*(t_1), \ldots, Z_{ib}^*(t_M)),$$

con media $\mathbf{0}$ y matriz de covarianza $(\hat{K}_i(t_p, t_q))_{1 \leq p,q \leq M}$, para cada $i = 1, \ldots, I$. Que a su vez utilizamos para calcular el valor simulado del estadístico $V$:

$$\tilde{V}_b = \sum_{i<j}^{I} \|Z_{ib}^* - C_{ij} Z_{jb}^*\|^2.$$

### Cálculo del p-valor

Hemos obtenido la distribución empírica de $V$ bajo la hipótesis nula $\{\tilde{V}_b\}_{b=1}^B$. Es sencillo calcular un p-valor aproximado mediante un procedimiento Monte Carlo:

$$p_{valor} = P(V > V_n) \approx \frac{\#\{\tilde{V}_b > V_n\}_{b=1}^B}{B} = \frac{1}{B} \sum_{b=1}^{B} \mathbf{1}_{\{\tilde{V}_b > V_n\}} \tag{2.20}$$

Podemos medir la precisión de este método en función del número $B$ de muestras Monte Carlo que realicemos. Se hace una analogía entre el error del procedimiento y la desviación típica del $p_{valor}$. Esto resulta razonable, pues una desviación típica nula indicará que hemos alcanzado el p-valor exacto. La desviación típica en función de $B$ es, de hecho:

$$\sigma_p(B) = V\left(\frac{1}{B} \sum_{l=1}^{B} \mathbf{1}_{\{\tilde{V}_b > V_n\}}\right)^{1/2} = \frac{1}{2\sqrt{B}} \tag{2.21}$$

### 2.3.2. Análisis post-hoc

Imaginemos que, siguiendo el ejemplo del apartado anterior, hemos concluido nuestro contraste ANOVA rechazando la hipótesis nula (2.11). Este resultado es útil, pues ha servido para determinar que efectivamente existen diferencias significativas entre las temperaturas dependiendo del clima en el que nos encontremos. Aún así, es posible preguntarse entré qué climas en concreto se encuentran estas diferencias. Éste estudio se conoce como análisis a posteriori o *post-hoc*.

#### Corrección de Bonferroni

En este contexto debemos realizar $\binom{I}{2}$ contrastes entre pares de medias de manera simultánea. Por esta razón, el riesgo de cometer un error de tipo I se dispara. Es necesario corregir los niveles de significación individuales que utilizamos si queremos un nivel de significación $\alpha$ global. Existen distintas maneras de realizar esta corrección, algunas de ellas pueden consultarse en [14, Capítulo 2]. En este trabajo utilizaremos la corrección de Bonferroni. Este método consiste en escoger un nivel de signficación individual $\alpha/q$, siendo $q$ el número de hipótesis nulas que debemos evaluar. De esta manera el nivel de significación global no superará $\alpha$.

#### Contraste $\mathrm{T}^2$

Cada una de las hipótesis nulas en un análisis *post-hoc* será:

$$H_0 : m_1(t) = m_2(t). \tag{2.22}$$

Nada impide que utilicemos el contraste ANOVA que se describió en el anterior capítulo para realizar esta prueba. Sin embargo, en esta sección presentaremos un contraste alternativo, un contraste específico de igualdad de medias entre dos muestras.

Cuando trabajamos sobre $\mathbb{R}$ el contraste de medias usual es el de la $t$ de Student [20, Capítulo 7]. El contraste que presentaremos aquí se muestra en el artículo de Pini, Stamm y Vantini [10], que se basa en el estadístico $T^2$ de Hotelling, una generalización de la $t$ de Student.

Supongamos una muestra de $n$ variables $\{X_i\}_{i=1}^n$ iid. que toman valores en $\mathbb{R}^p$. Sea $\mu$ su media, y $\Sigma$ su matriz de covarianza. Se define $T^2$ como

$$T^2 = n(\bar{X} - \mu)^\top \hat{\Sigma}^{-1}(\bar{X} - \mu), \tag{2.23}$$

donde $\bar{X}$ es el promedio y $\Sigma_n = \frac{1}{n-1}\sum_{i=1}^n (X_i - \bar{X})(X_i - \bar{X})^\top$ la matriz de covarianza muestral.

El estadístico mide la distancia entre ambas variables aleatorias ($\bar{X}$ y $\mu$), aunque no se trata de la distancia euclídea, porque tiene también en cuenta la correlación de la muestra. A este tipo de distancia se le conoce como distancia de Mahalanobis [13].

Para definir $T^2$ en el caso funcional debemos tomar otro camino, pues un operador de covarianza puede no ser invertible. En concreto, vamos a estudiar la propuesta que aparece en el artículo de Pini, Stamm y Vantini [16].

**Nota:** A partir de ahora omitiremos la variable $(t)$ de la notación, para facilitar la lectura de las ecuaciones. Para distinguir un elemento funcional de un vector utilizaremos la negrita para estos últimos.

Sean $X_1, \ldots, X_n$ variables aleatorias iid. evaluadas en $\mathbb{H}$, con media $m$ y operador de covarianza $\mathcal{K}$. Sea $\mathcal{D}_n = (\bar{X} - m) \otimes (\bar{X} - m)^\top$ el operador de error cuadrático medio. El operador tensorial $\otimes$ se definió en la Ecuación (2.10).

La $T^2$ de Hotelling en espacios de Hilbert se define como:

$$T^2 = n \max_{f \in \mathrm{Im}(\mathcal{K}_n) \setminus \{0\}} \frac{\langle f, \mathcal{D}_n f \rangle}{\langle f, \mathcal{K}_n f \rangle}, \tag{2.24}$$

El siguiente teorema nos proporciona una manera de calcular un valor aproximado de $T^2$ matricialmente.

**Teorema 2.2.** *Sea $\mathbb{H}$ un espacio de Hilbert separable y $\{V_p\}_{p \geq 1}$ una sucesión de subespacios tales que $V_p \subset \mathbb{H}, \dim(V_p) = p$ y $\lim_{p \to \infty} \inf_{\omega_p \in V_p} \|h - \omega_p\| = 0$, para todo $h \in \mathbb{H}$. Sea $x_1, \ldots, x_n$ una muestra de $n$ variables aleatorias iid. que toman valores en $\mathbb{H}$, con media $m$ y operador de covarianza $\mathcal{K}$, tales que $E_{\mathbb{R}}(\|X_1\|^2) < \infty$. Entonces, la sucesión $\{T_p^2\}_{p \geq 1}$ definida por*

$$T_p^2 = n \max_{f \in \mathrm{Im}(\mathcal{K}_n) \cap V_p \setminus \{0\}} \frac{\langle f, D_n f \rangle}{\langle f, \mathcal{K}_n f \rangle}, \tag{2.25}$$

*tiene las siguientes propiedades:*

*Si $(e_1, \ldots, e_p)$ es una base de $V_p$ y $W$ la matriz simétrica e invertible $p \times p$, tal que $W_{jk} = \langle e_j, e_k \rangle$, y sean las representaciones en esta base de $x_i$, $m$, $\hat{X}$ y $\hat{\Sigma}$*

$$\mathbf{x}_i = \mathbf{W}^{-1}(\langle x_i, e_1 \rangle, \ldots, \langle x_i, e_p \rangle)^\top, \quad \hat{\mathbf{m}} = \mathbf{W}^{-1}(\langle \bar{X}, e_1 \rangle, \ldots, \langle \bar{X}, e_1 \rangle)^\top,$$

$$\mathbf{m} = \mathbf{W}^{-1}(\langle m, e_1 \rangle, \ldots, \langle m, e_1 \rangle)^\top, \quad \hat{\mathbf{\Sigma}} = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x}_i - \hat{\mathbf{m}})(\mathbf{x}_i - \hat{\mathbf{m}})^\top.$$

*Entonces,*

$$T_p^2 = n(\hat{\mathbf{m}} - \mathbf{m})^\top \mathbf{W}^{1/2} (\mathbf{W}^{1/2} \hat{\mathbf{\Sigma}} \mathbf{W}^{1/2})^+ \mathbf{W}^{1/2} (\hat{\mathbf{m}} - \mathbf{m}). \tag{2.26}$$

*Además, $T_p^2 \xrightarrow{c.s.} T^2$, cuando $p \to \infty$.*

Donde $A^+$ denota la pseudo-inversa de Moore-Penrose del operador $A$ [2].

[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.pinv.html

Como se ha comentado en la Sección 2.1.2 los datos funcionales se representarán en muchas ocasiones a través de una base de funciones finita. Éste teorema nos permite calcular el valor de $T^2$ en función de los vectores de coeficientes de la representación en bases de los elementos de (2.25) y de la matriz de Gram de la base.

Llegados a este punto podemos adaptar el estadístico $T^2$ para que mida la distancia entre dos medias poblacionales. De este modo podremos utilizarlo para probar $H_0$ (2.22).

Definimos el estadístico del test bajo la hipótesis nula de la siguiente manera:

$$T_0^2 = \left(\frac{1}{n_1} + \frac{1}{n_2}\right)^{-1} \max_{f \in \mathrm{Im}(\mathcal{K}_{n_{\text{pooled}}}) \cap V_p \setminus \{0\}} \frac{\langle f, \mathcal{D}_{n_0} f\rangle}{\langle f, \mathcal{K}_{n_{\text{pooled}}} f\rangle},$$

donde $\mathcal{D}_{n_0} = (\bar{X}_1 - \bar{X}_2) \otimes (\bar{X}_1 - \bar{X}_2)$, y $\mathcal{K}_{n_{\text{pooled}}}$ se define como

$$\mathcal{K}_{n_{\text{pooled}}} : \mathbb{H} \to \mathbb{H} : f \mapsto \frac{n_1 - 1}{n_1 + n_2 - 2}(\mathcal{K}_{n_1} f) + \frac{n_2 - 1}{n_1 + n_2 - 2}(\mathcal{K}_{n_2} f).$$

Por cómo hemos definido $\mathcal{D}_{n0}$ queda patente que en el numerador estamos midiendo la distancia entre ambas medias muestrales, mientras que en el denominador se tiene en cuenta una combinación lineal de las covarianzas. Por lo visto en el Teorema 2.2 podemos calcular $T_0^2$ de forma matricial fijada una base finita de un subespacio de dimensión $p$ de $\mathbb{H}$ con matriz de Gram $\mathbf{W}$:

$$T_{0p}^2 = \left(\frac{1}{n_1} + \frac{1}{n_2}\right)^{-1} (\hat{\mathbf{m}}_1 - \hat{\mathbf{m}}_2)^\top \mathbf{W}^{1/2} (\mathbf{W}^{1/2} \hat{\Sigma}_{pooled} \mathbf{W}^{1/2})^{+} \mathbf{W}^{1/2} (\hat{\mathbf{m}}_1 - \hat{\mathbf{m}}_2). \tag{2.27}$$

En este caso ejecutaremos un test basado en permutaciones para obtener el p-valor del contraste. Nos encontramos en el caso homocedástico (mismo operador de covarianza $\mathcal{K}$), por lo que bajo $H_0$ todos los datos han sido extraídos de variables aleatorias idénticamente distribuidas. El test basado en permutaciones consiste en evaluar el estadístico $T_{0p}^2$ sobre todas las posibles combinaciones de los $n_1+n_2$ datos, tomadas de $n_1$ en $n_1$. En total hay un número $B = (n_1+n_2)!/(n_1!n_2!)$ de reordenaciones posibles. Este proceso se esquematiza en el Algoritmo 2.5.

De esta manera, el p-valor se calcula como:

$$p_{valor} = \frac{1}{B} \sum_{b=1}^{B} \mathbf{1}_{\{T_0^2(x^*_{1_b}, x^*_{2_b}, \ldots, x^*_{n_b}) \geq T_0^2(x_1, x_2, \ldots, x_n)\}}, \tag{2.28}$$

donde $(x^*_{1_b}, x^*_{2_b}, \ldots, x^*_{n_b})$ representa la b-ésima reordenación de la muestra.

```
Input: Muestras de datos – fd1, fd2
Output: Distribución empírica - dist

i ← 0
for c in Combinaciones(Size(fd1) + Size(fd2), Size(fd1)) do
      fd1_r, fd2_r ← Reordenar(fd1, fd2, c)
      dist[i++] ← EstadisticoT20(fd1_r, fd2_r)
end
devolver(dist)
```

**Figura 2.5:** Algoritmo del test basado en permutaciones. Para cada reordenación de los datos en dos muestras de tamaño fijo igual al de las muestras originales se calcula el valor del estadístico $T_0^2$.

# 3

# Análisis, diseño y desarrollo

El proyecto `scikit-fda` se lleva desarrollando desde el año 2018. Comenzó como el trabajo de fin de grado del alumno Miguel Carbajo [1], para después pasar a cargo de Carlos Ramos, del Grupo de Aprendizaje Automático de la Escuela Politécnica Superior. Desde entonces ha contado con múltiples aportaciones de alumnos que centran su proyecto de fin de grado en ampliar la funcionalidad de la librería, como en este caso.

A día de hoy el proyecto se encuentra en una fase más avanzada, donde gran parte de la funcionalidad básica ya está implementada. Actualmente el proyecto cuenta con cerca de mil descargas mensuales en PyPI, y con más de 40 estrellas en la plataforma Github [9]. Además, algunos usuarios ajenos a la Escuela contribuyen a su desarrollo reportando bugs o sugerencias. Esto es gracias a los estándares de codificación y herramientas que facilitan y agilizan el trabajo colaborativo y ayudan a integrarlo dentro de otros paquetes de uso muy extendido.

En esta sección analizaremos todos estos aspectos. Comenzaremos por el análisis de requisitos que se hizo al comienzo del proyecto, y que se encuentran en continua revisión. Veremos también cómo es el diseño de la librería, en términos de módulos y de interfaz para el usuario. Por último, repasaremos las tecnologías que sirven de soporte para el proyecto.

## 3.1. Análisis de requisitos

En esta sección se citan los requisitos no funcionales básicos sobre los que se basa el proyecto. La mayoría de ellos se fijaron al comienzo del desarrollo, sin embargo otros se han añadido durante la producción, fruto de una constante revisión de las necesidades de la comunidad `Python` y de las nuevas tecnologías disponibles. La finalidad de esto es mantener una librería actualizada y consistente con los estándares vigentes.

- El proyecto debe ser de código abierto.
- El software debe desarrollarse en lenguaje `Python`.
- Se debe seguir el estándar PEP 8 para la codificación y el PEP 257 para la documentación.
- La documentación debe ser detallada, incluyendo en ella ejemplos de uso de cada funcionalidad.

- El software debe ser escalable y debe permitir de manera sencilla contribuciones de la comunidad.
- El proyecto debe incluir tests unitarios y mecanismos de integración continua.
- La programación de la librería debe orientarse hacia la eficiencia, dando preferencia a algoritmos ya existentes en librerías como *numpy*, *scipy* o *scikit-learn*, pre-codificados en lenguajes de bajo nivel.
- El software debe poder ejecutarse, al menos, en los sistemas operativos Linux, MacOs y Windows.
- La Interfaz de programación de aplicaciones (API) debe parecerse en la medida de lo posible a la de *numpy*, *scipy* y *scikit-learn*.

## 3.2. Diseño

La librería está estructurada en los módulos que aparecen en la Figura 3.1. Al dar comienzo a este trabajo de fin de grado todavía no se había iniciado la implementación del módulo de inferencia estadística, siendo los contrastes que hemos tratado en este texto las primeras aportaciones al mismo.

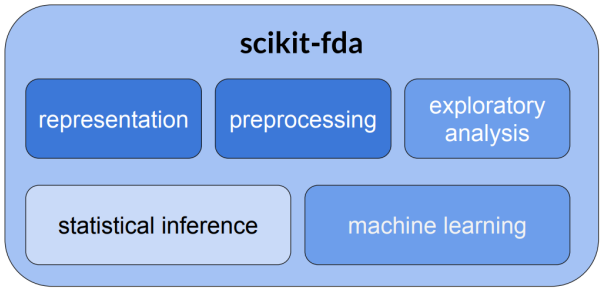



**Figura 3.1:** Módulos principales que componen la librería scikit-fda. La intensidad del color representa la cantidad de funcionalidad implementada en cada módulo al inicio de este trabajo.

A continuación repasaremos brevemente la funcionalidad principal que ofrece cada uno de los módulos de la librería, haciendo especial hincapié en la que se ha añadido a lo largo de este trabajo.

### 3.2.1. Representación

En este módulo se recogen las herramientas básicas para el manejo de datos funcionales, que se implementan alrededor de la clase abstracta `FData`. Existen dos tipos de instancia para esta clase, por un lado `FDataGrid`, para datos funcionales discretizados, y por otro lado `FDataBasis`, para representaciones en bases. También se proporcionan clases específicas para el manejo de bases, como `Fourier` o `BSpline`.

### 3.2.2. Preprocesamiento

Incluye funcionalidad básica para suavizar o alinear los datos de la muestra, entre otras técnicas. Paralelamente a este trabajo el alumno Yujian Hong ha incluido en este módulo herramientas de reduc-

ción de dimensionalidad, en concreto una técnica de análisis de componentes principales funcionales.

### 3.2.3. Análisis exploratorio

Se centra en tres puntos: medias de profundidad, detección de outliers y visualización de datos. Este último submódulo es de especial interés, pues ha sido de utilidad para la representación de los ejemplos y simulaciones que se muestran en el Capítulo 4.

### 3.2.4. Aprendizaje automático

Incluye herramientas de clasificación, clustering y regresión sobre datos funcionales.

### 3.2.5. Inferencia

Es el módulo principal sobre el que se ha desarrollado este trabajo.

Contiene la implementación de los contrastes ANOVA de un factor y $T^2$ de Hotelling, que se han presentado en el Capítulo 2, en las funciones `oneway_anova` y `hotelling_test_ind` respectivamente. Además se facilitan al usuario los estadísticos utilizados en cada test de manera independiente: `v_sample_stat` (2.17), `v_asymptotic_stat` (2.18), `t2_hotelling` (2.27).

Para diseñar la API de estos procedimientos se ha tomado como referencia la de algunas funciones similares, que ya existen en el ecosistema `Python`, en concreto `scipy.stats.f_oneway` [1], que es útil para ejecutar un contraste ANOVA clásico.

En esta función de se permite indicar un número variable de argumentos de entrada, donde cada uno corresponde a la muestra de un nivel del factor. Este mismo formato se ha utilizado en la librería `scikit-fda`. En el caso clásico cada muestra es una colección de valores, mientras que en el caso funcional es un objeto `FDataGrid`, o un `FDataBasis`. Se mantiene también el mismo retorno, en este caso el valor del estadístico $V_n$ y el p-valor del contraste.

Se incluyen otros parámetros opcionales que permiten al usuario configurar varios aspectos de la ejecución:

- *equal_var*: Flag que indica si el test se debe ejecutar en el caso homocedástico o heterocedástico. Por defecto se considera el caso homocedástico.
- *n_reps*: Número de muestras Monte Carlo a generar para la aproximación del p-valor. Por defecto se ha escogido un valor de 2000, que proporciona un error de precisión del orden de $10^{-2}$.
- *p*: Se da la opción de cambiar la norma $L^p$ que se utiliza en los estadísticos.
- *return_dist*: Flag opcional que permite devolver, como tercera variable de retorno, un array de numpy con todas las muestras Monte Carlo que se han simulado. Por defecto está desactivado.

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.f_oneway.html

- *random_state*: Parámetro de uso extendido en la librería `numpy` y `scikit-learn` que permite fijar una semilla para obtener resultados reproducibles en procedimientos que contienen generación de valores aleatorios, como en este caso la simulación de los procesos gaussianos.

Para el test de la $T^2$ de Hotelling (`hotelling_test_ind`) se ha tomado como referencia la función `scipy.stats.ttest_ind` [2], que implementa el test de la $t$ de Student clásico. En este caso se reciben dos muestras de datos, y se devuelven de nuevo el p-valor y el valor del estadístico. Existe una versión alternativa de la función, `scipy.stats.ttest_ind.from_stats` [3], que recibe las medias y varianzas muestrales en lugar del total de la muestra. Sin embargo, dado que el test implementado en el caso funcional se basa en permutaciones de los datos este formato no aplica. Como el número de combinaciones a evaluar puede ser muy elevado se ha tomado la decisión de permitir al usuario escoger un número máximo de iteraciones en el algoritmo. En cada iteración se evaluará una combinación aleatoria. Para configurar este comportamiento se utilizan los siguientes parámetros opcionales:

- *n_reps*: Número máximo de reordenaciones aleatorias de los datos a aplicar en el test de permutaciones.
- *return_dist*: Flag opcional que permite devolver, como tercera variable de retorno, un array de numpy con todas los estadíticos calculados a partir de las permutaciones que se han evaluado.
- *random_state*: Para obtener resultados reproducibles al generar las combinaciones de manera aleatoria.

### 3.2.6. Otros módulos

Existen otros módulos con utilidad auxiliar. Uno de ellos es el módulo de datasets. En él se puede encontrar funcionalidad para la generación de datos sintéticos (por ejemplo, la función que permite la simulación de procesos gaussianos, `make_gaussian_process`) y para importar conjuntos de datos reales. Muchos de estos datos se obtienen directamente de las librerías de FDA de referencia en `R`, como `fda` y `fda.usc`.

También merece la pena mencionar el módulo misceláneo, que recoge, por ejemplo, el submódulo de métricas, donde se han implementado diferentes normas y distancias para objetos funcionales, el de covarianzas, que integra los kernels de `scikit-learn` para su uso como funciones de covarianza de datos funcionales. En este trabajo se ha incluido la covarianza de ruido blanco `WhiteNoise`, que se utilizará en varios ejemplos del Capítulo 4.

## 3.3. Codificación, documentación y pruebas

La codificación del proyecto se ha realizado íntegramente en `Python`, siguiendo el estándar *PEP 8* [8], creado por Guido van Rossum. Su autor defiende que "un código es leído muchas más veces de

[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html
[3] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind_from_stats.html

lo que es escrito", por lo que propone esta convención para mejorar la legibilidad del mismo y hacerlo consistente para todos los usuarios de la comunidad. En esta guía de estilo se detallan pautas como el número de espacios al identar (cuatro en este caso), o la convención para los nombres de las variables y las clases.

También se ha seguido el estándar *PEP 257* [4] a la hora de crear la documentación [4]. Cada entidad, como las funciones, métodos o clases son anotadas con un comentario *docstring*. Existen distintas convenciones para la escritura de *docstrings*. En `scikit-fda` se ha optado por el estilo Google, que dispone de una sintáxis simple y clara. Este tipo de comentarios pueden ser exportados a PDF o HTML a través de la herramienta `Sphinx`, y pueden incluir fórumulas en LaTeX, así como ejemplos de uso, que sirven a su vez como tests.

Además de estos test integrados en los *docstrings*, conocidos como *doctest*, se incluyen tests unitarios exhaustivos para cada funcionalidad. También se han incluido ejemplos ilustrativos en formato notebook para las principales herramientas que contiene la librería. Estos ejemplos se pueden encontrar también en la documentación, ver Apéndice B.

## 3.4. Desarrollo

En esta sección revisaremos la metodología seguida durante la realización del trabajo dentro del proyecto.

Como principal forma de comunicación entre el equipo de `scikit-fda` se mantienen reuniones semanales de dos horas de duración. En ellas se expone un breve informe relacionado con la actividad que se ha llevado a cabo a lo largo de la semana, incluyendo sugerencias, detalles de implementación y los resultados obtenidos. Las reuniones se han mantenido de manera tanto presencial como telemática a lo largo de todo el curso.

La principal herramienta de desarrollo utilizada en el proyecto es *Github*. Esta plataforma integra el servicio de control de versiones *Git*, y permite almacenar en un servidor remoto los cambios en el código que realiza cada programador. Existen múltiples maneras de gestionar un sistema de control de versiones de manera eficiente y ordenada. En este proyecto se ha escogido un flujo de trabajo *Gitflow*. *Gitflow* detalla cómo debe ser el proceso de ramificación alrededor de las distintas versiones del código, de modo que se diferencien versiones en diferentes estados (estable, en producción, etc.).

La rama de desarrollo *develop* sirve de base ramas de función o *feature*, en las que se desarrollan nuevas funcionalidades. Por otro lado, en la rama maestra o *master* se incluyen las versiones estables en producción. Esta rama suele estar protegida y no se deberían efectuar cambios directamente sobre ella. El modo de pasar una versión que se considera estable de *develop* a *master* es a través de una

[4] https://fda.readthedocs.io/en/latest/index.html

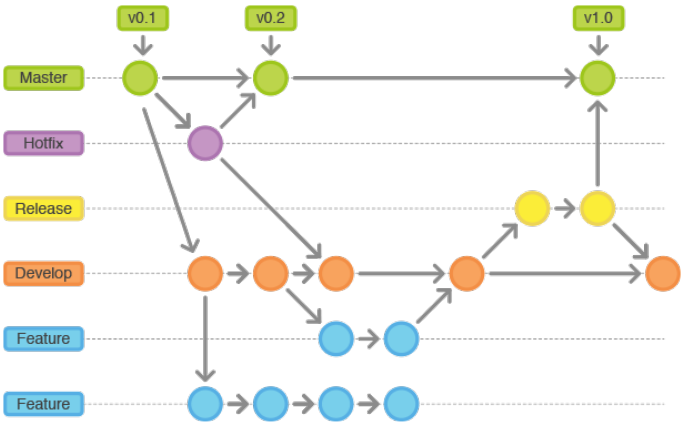



**Figura 3.2:** Ramificación de un proyecto basado en Gitflow. La rama Master es la que contiene la versiones estables, mientras que la rama Develop es la base para el desarrollo de nueva funcionalidad.

rama *release*. Por otro lado, en caso de necesitar solucionar un bug que se ha encontrado en las ramas develop o máster existen un tipo de ramas auxiliares, llamadas *hotfix*. Todo este flujo de trabajo puede verse reflejado en la Figura 3.2.

*Github* dispone de diferentes herramientas que complementan al sistema de versiones *Git*, como la creación de incidencias o *issues* donde se reportan posibles bugs o sugerencias para un desarrollo futuro. Otro beneficio que ofrece *Github* es la posibilidad de integrar componentes externos.

En este proyecto se utiliza el componente *Travis CI*. Se trata de una herramienta de integración continua. Esta técnica es de uso extendido en el desarrollo de software. Su objetivo es conseguir detectar errores en el proyecto lo antes posible, realizando pruebas de compilación y corriendo los tests cada vez que se realiza una modificación en alguna de las ramas. La herramienta *Travis* además permite realizar estas comprobaciones sobre distintas configuraciones, pudiendo escoger diferentes entornos, versiones del software y múltiples sistemas operativos. También es de utilidad la herramienta *Codecov*, que permite conocer qué porcentaje de la funcionalidad implementada ha sido cubierta con tests. Por último, la herramienta *readthedocs* permite mantener actualizada la documentación del proyecto con la última versión disponible de develop en *Github*.

# 4

# Resultados

En esta sección vamos a comprobar el funcionamiento de los procedimientos que se han ejecutado a lo largo de este trabajo. Utilizaremos diferentes conjuntos de datos, tanto sintéticos como reales, a los que se puede tener acceso a través de la librería `scikit-fda`. En el Apéndice A se puede encontrar un notebook de `Python`, con todo el código necesario para generar las gráficas y los resultados de esta sección, así como algunas pruebas complementarias.

## 4.1. Estudio con datos sintéticos

En este estudio realizaremos pruebas para comprobar si el funcionamiento de los contrastes que se han implementado es el esperado. Queremos comprobar que el p-valor que se obtiene en ambos métodos es más grande a medida que la diferencia entre las medias de cada nivel del factor es menos evidente.

Para ello utilizaremos datos sintéticos basados en el modelo lineal unifactorial:

$$X_{ij}(t) = m_i(t) + e_{ij}(t).$$

Por simplicidad escogemos tres niveles del supuesto factor ($I = 3$). Para cada nivel asignamos una función de media diferente, que podemos ver representadas en la Figura 4.1.

$$m_i(t) = t^i(1-t)^{6-i},\ i = 1, 2, 3.$$

Los residuos $e_{ij}(t)$ se modelan como trayectorias de procesos gaussianos $\mathcal{G}(0, K_w)$, donde $K_w$ es una función de covarianza de ruido blanco, que se define como:

$$K_w(s,t) = \begin{cases} \sigma^2, & s = t \\ 0, & s \neq t \end{cases}$$

donde el parámetro $\sigma^2$ representa la intensidad del ruido que agrega el residuo.

Generamos un total de 30 datos sintéticos, 10 para cada nivel del factor, muestreados en 100

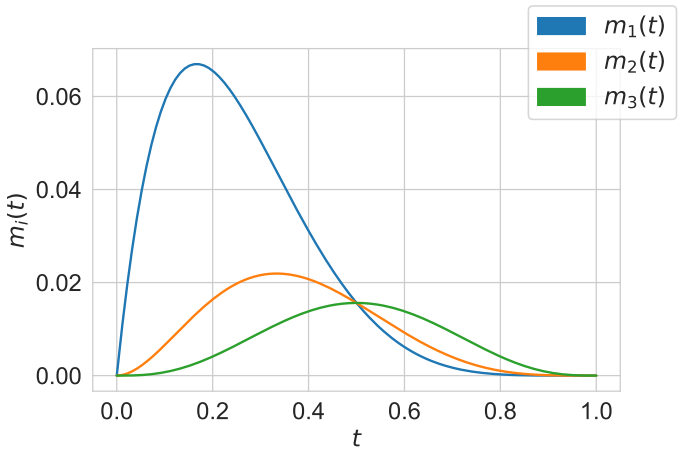



**Figura 4.1:** Medias base para cada nivel del factor de los datos sintéticos.

puntos equiespaciados del intervalo $[0, 1]$.

Con estos datos suponemos la hipótesis nula:

$$H_0 : m_1(t) = m_2(t) = m_3(t),$$

y las hipótesis nulas individuales:

$$H_{01} : m_1(t) = m_2(t), \quad H_{02} : m_2(t) = m_3(t), \quad H_{03} : m_1(t) = m_3(t).$$

Ejecutaremos el contraste ANOVA funcional para $H_0$ con diferentes niveles de ruido $\sigma^2$. Sobre las hipótesis nulas individuales aplicaremos ambos contrastes, el ANOVA y el $T^2$, para comparar el funcionamiento de ambos. Las intensidades de ruido que se utilizarán en cada repetición son $\sigma_1^2 = 10^{-3}\sigma_2^2 = 10^{-2}$, $\sigma_3^2 = 10^{-1}$ y $\sigma_4^2 = 1$.

| | $H_0$ | $H_{01}$ | | $H_{02}$ | | $H_{03}$ | |
|---|---|---|---|---|---|---|---|
| | ANOVA | ANOVA | $T^2$ | ANOVA | $T^2$ | ANOVA | $T^2$ |
| $\sigma_1^2$ | 0 | 0 | 0.001 | 0 | 0.003 | 0.296 | 0.201 |
| $\sigma_2^2$ | 0.07 | 0.107 | 0.166 | 0.077 | 0.342 | 0.418 | 0.218 |
| $\sigma_3^2$ | 0.358 | 0.424 | 0.427 | 0.310 | 0.590 | 0.416 | 0.205 |
| $\sigma_4^2$ | 0.433 | 0.521 | 0.517 | 0.355 | 0.629 | 0.412 | 0.199 |

**Tabla 4.1:** P-valores obtenidos para diferentes intensidades del parámetro de ruido $\sigma^2$, sobre la hipótesis múltiple $H_0$ aplicando el test ANOVA , y las hipótesis individuales utilizando tanto el método ANOVA como el test $T^2$.

Para interpretar los resultados (Tabla 4.1) nos centraremos primero en los p-valores del contraste de la hipótesis $H_0$. Como se puede observar, fijado un $\alpha = 0{,}05$ únicamente rechazamos la hipótesis nula para el nivel de ruido más bajo $\sigma_1^2$. Además, los p-valores crecen a medida que se incrementa la cantidad de ruido. Este resultado es el esperado, pues a mayor ruido es más difícil distinguir las

diferencias entre las medias y, por tanto, es más difícil rechazar la hipótesis de igualdad. Esta propiedad se debe ver reflejada tanto en los p-valores de $H_0$ como en el del resto de hipótesis individuales.

Veamos ahora los resultados obtenidos para $H_{01}$. En este caso estamos probando la igualdad entre $m_1(t)$ y $m_2(t)$. Los p-valores obtenidos son muy similares para ambos tests. Además, de nuevo, el p-valor crece a medida que lo hace la intensidad del ruido.

Sin embargo, para $H_{02}$ los p-valores difieren entre si. Ambos métodos rechazan la igualdad entre $m_1(t)$ y $m_3(t)$ para $\sigma_1^2$. Sin embargo observamos en el resto de niveles de ruido que $T^2$ arroja p-valores más altos que los correspondientes para el ANOVA funcional, a pesar de que crecen en la misma proporción. Esto se debe a que la manera en que se mide la diferencia entre las medias en ambos métodos es distinta, por lo que la $T^2$ penalizará más algunas situaciones concretas y viceversa.

Por último en $H_{03}$ se da un caso muy particular. Como se aprecia en la Figura 4.1 las diferencias entre $m_2(t)$ y $m_3(t)$ son de partida pequeñas. Por esta razón, al añadir ruido los p-valores saturan y no se distinguen más diferencias entre las medias. Esto se ve reflejado en la tabla. Se observa que para todas las intensidades de ruido $T^2$ obtiene un p-valor constante de 0.2. Por otro lado, el método ANOVA sí que puede cuantificar diferencia entre $\sigma_1^2$ y $\sigma_2^2$, pero en los sucesivos valores también se satura.

Desde un punto de vista general los resultados también son los esperados. Ambos contrastes detectan menos diferencias entre $m_2(t)$ y $m_3(t)$ que entre cualquiera de ellas con $m_1(t)$. De hecho, es la única hipótesis individual que no rechazaríamos en un análisis *post-hoc* con un nivel de ruido $\sigma_1^2$ y nivel de significación $\alpha = 0{,}05$.

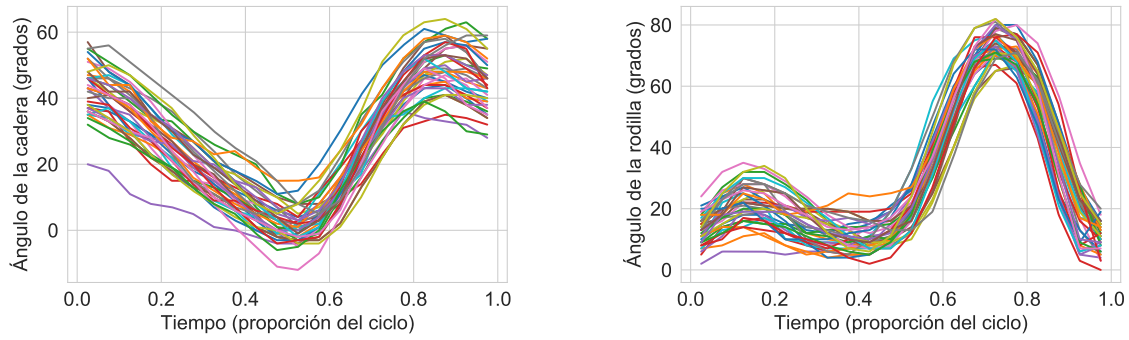
## 4.2. Convergencia del p-valor

En el contraste ANOVA aproximamos el p-valor realizando un número $B$ de pasos en el algoritmo. Como vimos, obtendremos el p-valor exacto del test cuando $B \to \infty$, por lo que siempre estaremos cometiendo un error de precisión para $B < \infty$. Ya definimos, en la Ecuación (2.21), una fórmula que nos permitiría estimar el error cometido al escoger un número $B$ finito de iteraciones, a través de la desviación típica del p-valor.

Por otro lado, como vimos en la Sección 3.2, no siempre será posible calcular todas las permutaciones posibles en el test $T^2$, al menos con recursos computacionales limitados, ya que, si tenemos dos muestras de tamaño $n_1$ y $n_2$ el número de combinaciones será $\binom{n_1+n_2}{n_1}$. Como alternativa se propone realizar un número $B <<< \binom{n_1+n_2}{n_1}$ de combinaciones aleatorias de los datos.

En este experimento se quiere observar de manera empírica cómo convergen estos métodos de aproximación a medida que $B$ crece. Esta prueba la realizaremos sobre el dataset "Gait". Este conjunto de datos fue obtenido por The Motion Analysis Laboratory at Children's Hospital, en San Diego, Estados

Unidos. Consta de 39 observaciones bivariadas funcionales. Cada una representa el ángulo formado por la rodilla y la cadera de un niño determinado al dar un paso, desde que levanta el pie que avanza hasta que éste toca el suelo. La duración de este ciclo se identifica con el intervalo $[0, 1]$. Las muestras se han tomado sobre 20 puntos equiespaciados de éste. Para una explicación más detallada de estos datos se puede consultar el artículo de Olshen et al. [15].




(a) Ángulo de la cadera.

(b) Ángulo de la rodilla.

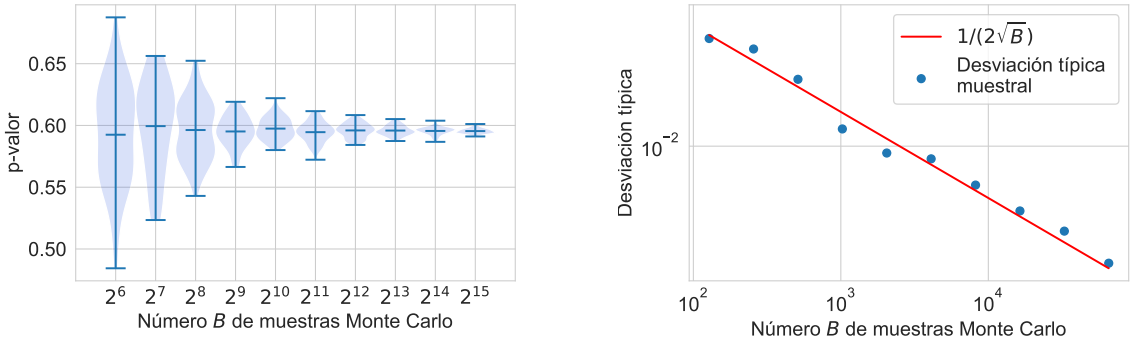
**Figura 4.2:** Ángulo formado por la cadera y la rodilla de 39 niños al dar un paso. A pesar de que en la imagen no es posible apreciarlo, existe una correspondencia entre cada trayectoria del ángulo de la cadera con otra del ángulo de la rodilla.

Para las pruebas utilizaremos solamente la variable de la rodilla, como muestra independiente. Se considerarán 3 niveles del factor aleatorios, es decir, escogemos al azar 13 trayectorias para el primero y 13 para el segundo. Las 13 restantes se asignarán al tercer nivel. Además, en este caso, los datos se mantendrán en su forma original discretizada.

Calcularemos 50 veces el p-valor de los contrastes, para cada $B$ en $\{2^i, i = 7, \ldots, 16\}$.

### 4.2.1. Contraste ANOVA

Ejecutamos el test ANOVA sobre los datos divididos en tres niveles, considerando la hipótesis nula $H_0 : m_1(t) = m_2(t) = m_3(t)$. Los resultados de este procedimiento se muestran en la Figura 4.3. A la izquierda se puede observar un gráfico de violín, donde se refleja la densidad de cada muestra de 50 p-valores, así como su máximo, mínimo y promedio. A simple vista se observa cómo la varianza de los datos decrece a medida que $B$ aumenta. A la derecha se ha representado la desviación típica de cada muestra junto a la gráfica de $\sigma_p(B) = 1/(2\sqrt{B})$. Vemos como los valores empíricos se acercan a los esperados, consiguiendo un ajuste con un error cuadrático medio del orden de $10^{-6}$.




(a) Gráfico de violín.

(b) Precisión observada y esperada.

**Figura 4.3:** En la figura (a) encontramos el diagrama de violín para muestras de 50 p-valores obtenidas a partir del método ANOVA funcional, con diferente número $B$ de muestras Monte Carlo. En la figura (b) se muestra la precisión obtenida en cada muestra (en azul) comparada con la esperada (en rojo). Para facilitar su visualización se ha utilizado la escala logarítmica.

### 4.2.2. Contraste $T^2$ de Hotelling

Para calcular el p-valor del test sobre la hipótesis nula $H_0 : m_1(t) = m_2(t)$ se requiere un total de $\binom{n_1+n_2}{n_1} = \binom{13+13}{13} \approx 10^7$ permutaciones. El valor máximo de $B$ que consideraremos en esta prueba es $2^{16} = 65536$. Cabe destacar que es posible que alguna de las permutaciones aleatorias se repita, pudiendo alterar el resultado obtenido. Este procedimiento por permutaciones aleatorias es de nuevo un método Monte Carlo, por lo que esperamos obtener una precisión del mismo orden que la del método ANOVA .

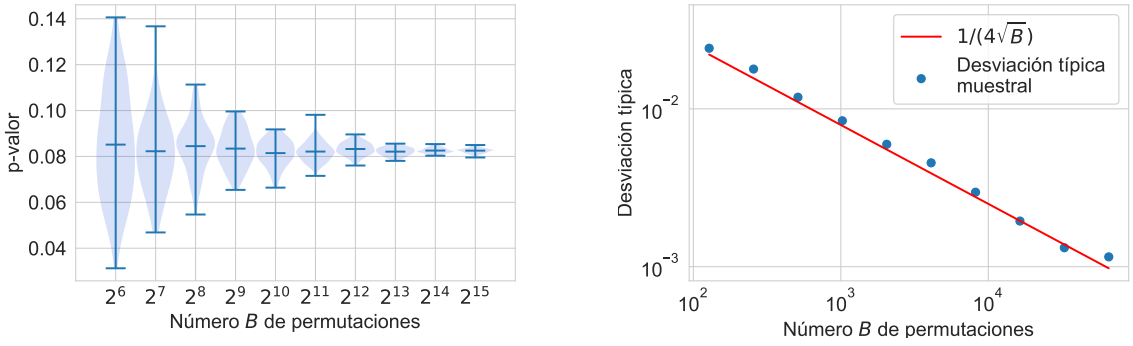
Los resultados que obtenemos de aplicar el mismo procedimiento se encuentran en la Figura 4.4.

Estos resultados se ajustan a lo esperado. Por un lado, observamos en el gráfico de violín (Figura 4.4(a)) cómo los p-valores registrados convergen a un valor. En la Figura 4.4(b) observamos que en esta ocasión la desviación típica del p-valor se ajusta con un error cuadrático medio de $10^{-5}$ a la curva $1/(4\sqrt{B})$, por lo que converge de manera similar al método ANOVA .

## 4.3. Estudio meteorológico de Canadá

En esta sección retomamos el ejemplo que vimos al comienzo de la Sección 2.3. Disponemos de un total de 15 trayectorias, que representan la temperatura media registrada en una estación meteorológica de Canadá a lo largo de un año. Como se aprecia en la Figura 2.3 cada trayectoria ha sido etiquetada con un clima determinado, dependiendo de la localización de la estación en cuestión.

Nuestro objetivo es conocer si el tipo de clima (el factor) ha influido sobre la temperatura media




(a) Gráfico de violín.

(b) Precisión observada.

**Figura 4.4:** En la figura (a) encontramos el diagrama de violín para muestras de 50 p-valores obtenidas a partir del método $T^2$ de Hotelling funcional, con diferente número $B$ de permutaciones aleatorias. En la figura (b) se muestra la precisión obtenida en cada muestra (en azul) comparada con la curva $1/(4\sqrt{B})$ (en rojo). Para facilitar su visualización se ha utilizado la escala logarítmica.

registrada (variable respuesta). Nos encontramos en un claro problema de diseño de experimentos, por lo que podemos aplicar la función `anova_oneway` que hemos implementado para conocer los resultados del contraste sobre la hipótesis nula:

$$H_0 : m_a(t) = m_c(t) = m_p(t),$$

donde $m_a(t)$ es la temperatura media en el clima ártico, $m_c(t)$ en el clima continental y $m_p(t)$ en el pacífico.

Para hacer más ligera la computación representamos los datos en una base de Fourier de 9 elementos (8 pares seno-coseno más la constante). La base de Fourier parece la indicada para este tipo de datos, ya que presentan cierta periodicidad (la temperatura del primer día del año debe ser similar a la del último). Por otro lado, como vemos en la Figura 4.5 con 9 elementos en la base ya queda reflejada la tendencia general de la temperatura a lo largo del año, que es lo que queremos contrastar. Podemos entender este paso como un preprocesado que elimina parte del ruido que representaba el residuo en el modelo.

A simple vista se observan diferencias notables entre las muestras de cada tipo de clima, por lo que lo esperado sería rechazar $H_0$ con un nivel de significación $\alpha = 0{,}05$. Se observa que la temperatura en el clima pacífico es suave, en el sentido de que tiene poca variación a lo largo del año pasando de los -10ºC de mínima a los 20ºC de máxima en verano. Por el contrario, la temperatura en el clima ártico es extrema, con inviernos muy fríos (-30ºC) y veranos templados, alcanzando en algunos lugares los 12ºC. La temperatura continental toma valores intermedios entre estos dos climas.

Si aplicamos el contraste ANOVA con 2000 simulaciones obtenemos un p-valor nulo. Por esa razón rechazamos $H_0$ sin problema y llegamos a la conclusión de que, efectivamente, el tipo de clima en el

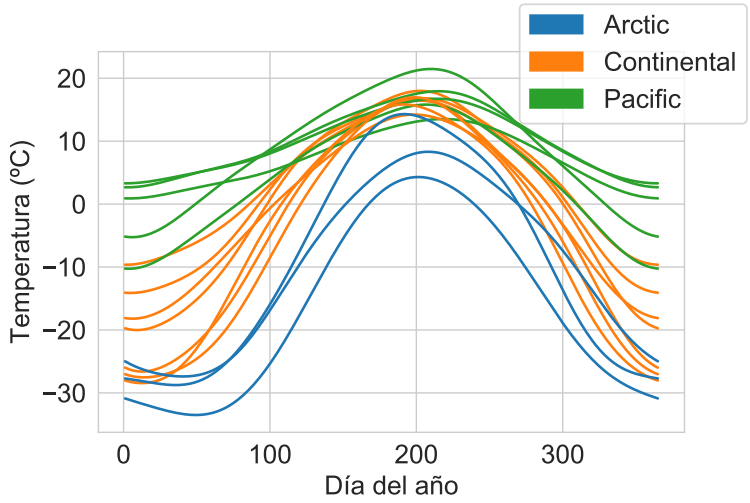



**Figura 4.5:** Trayectorias seleccionadas y etiquetadas según el clima. En azul se representan las temperaturas registradas en clima ártico, en naranja en clima continental y en verde en clima pacífico. Se ha escogido una representación en una base de Fourier de 9 elementos.

que se muestrea una temperatura influye en el valor obtenido notablemente.

Para completar nuestro estudio realizaremos un análisis *post-hoc*, ya que hemos rechazado $H_0$. Queremos saber entre qué pares de medias existen realmente diferencias. Planteamos las siguientes hipótesis nulas individuales:

$$H_{01} : m_a(t) = m_p(t), \quad H_{02} : m_a(t) = m_c(t), \quad H_{03} : m_c(t) = m_p(t).$$

Si queremos mantener un nivel de significación global para la prueba $\alpha = 0{,}05$ debemos utilizar un nivel de significación individual $\alpha_i \approx 0{,}017$. Aplicamos el análisis a posterior utilizando el test $T^2$, sobre todas las permutaciones posibles de los datos. Los resultados se muestran en la Tabla 4.2.

| | $H_{01}$ | $H_{02}$ | $H_{03}$ |
|---|---|---|---|
| p-valor | 0 | 0.025 | 0.206 |

**Tabla 4.2:** Resultados del análisis *post-hoc* para las hipótesis de igualdad entre las medias de las temperaturas del clima ártico y pacífico ($H_{01}$), ártico y continental ($H_{02}$) y continental y pacífico ($H_{03}$). Para obtener estos p-valores su utilizó el contraste de la $T^2$ de Hotelling.

Los resultados encajan con lo esperado. Podemos rechazar únicamente la igualdad entre las medias del clima ártico y el pacífico. Esto era evidente a simple vista. Por otro lado no hemos podido rechazar la igualdad entre las medias del clima continental con los otros dos. Este hecho también tiene sentido, pues algunas observaciones del clima continental son muy similares a las del pacífico, y lo mismo ocurre (en mayor medida) con las del clima ártico, donde el p-valor obtenido es bastante grande.

La conclusión que podemos extraer de este estudio es que el factor clima influye en las temperatu-

ras del clima ártico y el pacífico, mientras que entre el clima continental y el resto de climas la influencia del factor no es tan evidente.

## 4.4. Estudio meteorológico de España

A continuación realizaremos un estudio muy similar con datos meteorológicos de España. En concreto, utilizaremos un conjunto de datos extraídos de la página de la Agencia Estatal de Meteorología de España (AEMET) . Este *dataset* contiene registros de temperatura media, precipitaciones y velocidades del viento, tomadas entre los años 1980 y 2009 en 73 estaciones meteorológicas distribuidas por toda la geografía española. En este caso nos volveremos a centrar en la variable de la temperatura media anual, que se puede ver representada en la Figura 4.6.

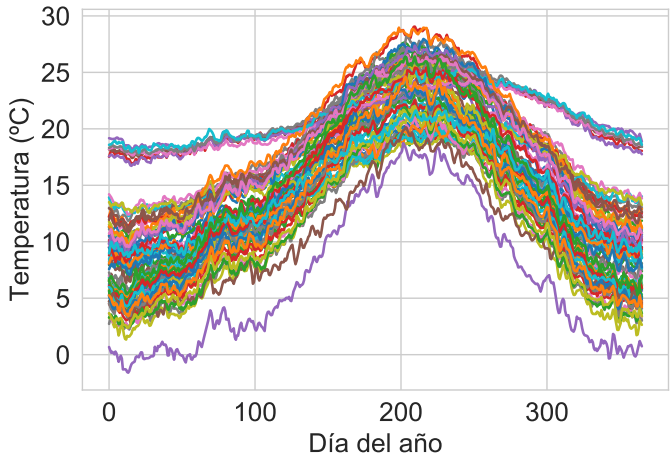
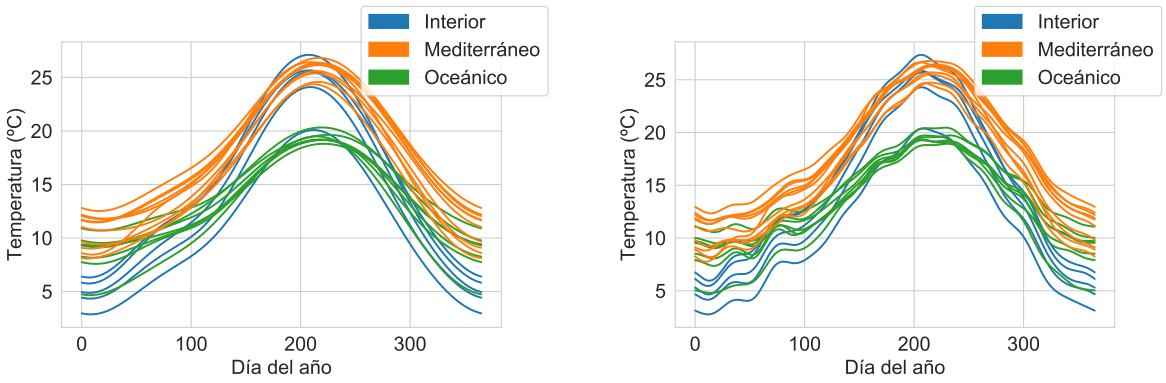



**Figura 4.6:** En la figura se muestran 73 trayectorias correspondientes a la temperatura media registrada en diferentes estaciones meteorológicas de España entre los años 1980 y 2009.

Éste conjunto de datos, a diferencia del anterior, no contiene etiquetas que indiquen el tipo de clima, por lo que se han añadido a mano estación por estación. Una vez etiquetados los datos procedemos a escoger un subconjunto de ellos para su estudio. En concreto se han extraído 22 registros de temperatura, 10 correspondientes al clima mediterráneo, 5 al clima mediterráneo de interior y 7 al clima oceánico. Los de datos seleccionados se representan en la Figura 4.7. En esta ocasión hemos escogido dos tipos de representaciones en bases diferentes. La primera, en la subfigura (a) se ha escogido una base de Fourier con un número de elementos $k = 9$. En la segunda (b), se ha utilizado $k = 25$ sobre la misma base. El objetivo de este estudio es ilustrar cómo influye la representación en bases escogida en los resultados obtenidos al contrastar las medias.

De nuevo queremos determinar si el factor tiene relevancia sobre la temperatura registrada. A simple vista podemos apreciar algunas diferencias generales entre los climas. Por ejemplo, el clima oceánico parece bastante similar al mediterráneo en términos de la amplitud del rango de temperatu-

ras. Ambas muestran unos 10ºC de diferencia entre los máximos de verano y los mínimos de invierno, sin embargo, a pesar de ser ambos climas suaves, el mediterráneo mantiene una temperatura más elevada que el oceánico, por lo que parece una réplica del mismo desplazada sobre el eje vertical. Por otro lado, tenemos el clima mediterráneo de interior, en el que observamos dos características relevantes, la primera es que la variación de la temperatura es mucho mayor a lo largo del año, llegando a diferencias de más de 20ºC entre el verano y el invierno. Además, se observa que la separación entre los registros de temperatura de este clima es mayor que en el resto.




(a) Representación en bases de Fourier con $k = 9$.

(b) Representación en bases de Fourier con $k = 25$.

**Figura 4.7:** En la figura se muestran 22 trayectorias correspondientes a la temperatura media registrada en diferentes estaciones meteorológicas de España entre los años 1980 y 2009. En azul se representan las correspondientes al clima mediterráneo de interior, en naranja al clima mediterráneo y en verde al clima oceánico. A la izquierda podemos ver los datos representados en una base de Fourier con 9 elementos, mientras que a la derecha se han utilizado 25. Se puede apreciar como una base de menor dimensión consigue una mayor suavidad en la representación de los datos.

Probamos, a través del test ANOVA funcional la hipótesis nula

$$H_0 : m_m(t) = m_i(t) = m_o(t),$$

donde $m_m(t), m_i(t), m_o(t)$ representan las medias para los climas mediterráneo, mediterráneo de interior y oceánico respectivamente.

Al igual que ocurría en el ejemplo anterior obtenemos un p-valor nulo en ambas representaciones, lo que nos permite rechazar $H_0$. Pasando ahora al análisis *post-hoc*, planteamos las siguientes hipótesis nulas individuales:

$$H_{01} : m_m(t) = m_i(t),\ H_{02} : m_m(t) = m_o(t),\ H_{03} : m_i(t) = m_o(t).$$

Aplicamos la corrección de Bonferroni para obtener un nivel de significación individual. Para un $\alpha = 0{,}05$ global tomamos un $\alpha_i \approx 0{,}017$ individual. En la Tabla 4.3 se muestran los resultados del test $T^2$ de Hotelling.

| | $H_{01}$ | $H_{02}$ | $H_{03}$ |
|---|---|---|---|
| $k = 9$ | 0.005 | 0 | 0.039 |
| $k = 25$ | 0.002 | 0 | 0.003 |

**Tabla 4.3:** Resultados del análisis *post-hoc* para los datos representados en bases de Fourier con $k = 9$ y $k = 25$. Se comparan las medias de las temperaturas del clima mediterráneo y mediterráneo de interior ($H_{01}$), mediterráneo y oceánico ($H_{02}$) y mediterráneo de interior y oceánico ($H_{03}$). Para obtener estos p-valores su utilizó el contraste de la $T^2$ de Hotelling.

En la representación más suavizada, con $k = 9$ se rechaza la igualdad entre los climas mediterráneo y mediterráneo de interior, y entre el clima mediterráneo y oceánico. Se encuentra mayor diferencia entre estos últimos, ya que el p-valor es nulo. Éste resultado es razonable, pues en la Figura 4.7(a) se aprecia que ambas submuestras están notablemente distanciadas. Sin embargo, no rechazamos la igualdad entre el clima mediterráneo de interior y el oceánico. Esto puede deberse a la influencia del outlier que presenta el clima interior, que no alcanza temperaturas elevadas en verano.

En el caso en el que $k = 25$ sin embargo sí que rechazamos todas las hipótesis. Al considerar una representación más detallada de los datos de la muestra entran en juego nuevas características, que aportan más información, lo que permite en rechazar la igualdad de medias. En este ejemplo en concreto se aprecia que todos los registros del clima mediterráneo de interior presentan un pico característico en el verano, mientras que en el clima oceánico se observa una meseta. Esta característica, que no se reflejaba en la representación con $k = 9$ ha sido uno de los factores que ha influido en que en esta ocasión el p-valor haya sido suficientemente bajo.

Este resultado subraya la importancia de seleccionar una base adecuada para la representación de los datos. Si queremos estudiar la tendencia de la temperatura a grandes rasgos una base con pocos elementos puede ser la indicada, pero debemos ser conscientes de que perderemos algunas características e información de los datos que pueden resultar clave.

# 5

# Conclusiones

A lo largo de este trabajo de fin de grado se ha detallado el proceso de incluir nuevas herramientas de FDA en la librería `scikit-fda`. Se ha puesto especial atención en facilitar una solución robusta y eficiente, centrada en las necesidades de la comunidad `Python`. La accesibilidad y ergonomía que proporciona este lenguaje ha provocado que su uso se haya extendido en los últimos años y que muchas de las tecnologías emergentes, sobre todo en campos como la Inteligencia Artificial y la Ciencia de Datos, lo hayan utilizado como base. Se ha querido seguir esta filosofía para acercar algunos métodos del FDA a los usuarios de `Python`. Es importante subrayar que el proyecto es por completo *open source*, permitiendo la colaboración de cualquier programador que desee realizar aportaciones.

Como hemos detallado en el Capítulo 3, se ha participado en múltiples fases del desarrollo de software, tales como diseño, codificación, pruebas y las primeras fases del mantenimiento. Además, se han utilizado tecnologías específicas de aseguramiento de la calidad e integración continua como *Github*, *Codecov* o *Travis*, que suponen una herramienta indispensable y cada vez más implantada en los proyectos de software.

El trabajo se podría ampliar al menos por dos vías diferentes. Por un lado existen múltiples contrastes estadísticos y herramientas de FDA que se pueden añadir a la librería `scikit-fda`. Incluso existen otras alternativas a las implementaciones de los tests aquí estudiados, que podrían compararse en términos de eficiencia y velocidad de convergencia. Por otro lado, podría ser muy interesante encontrar un problema con datos funcionales reales sobre el que hacer un estudio en profundidad, utilizando la librería y en concreto las técnicas que aquí se han desarrollado.

Una componente importante en el trabajo ha sido la extracción e interpretación de resultados, fruto de analizar datos reales y sintéticos con las técnicas implementadas. Con esto se pretendía validar el correcto funcionamiento de las herramientas, para posteriormente poder utilizarlas en el análisis de datos. Por esta razón se ha ejercitado mucho la capacidad de comunicación a la hora de transmitir estos resultados y el trabajo realizado a través de informes y reuniones semanales. El trabajo en equipo ha sido una competencia que se ha desarrollando a lo largo de todo el grado, y que especialmente en este proyecto ha tenido una importancia clave. Cualquier profesión relacionada con la Informática va a requerir de estas capacidades de comunicación y trabajo en equipo, por lo que enfocar el trabajo

desde un punto de vista colaborativo es un aspecto muy positivo.

Por otro lado, no hay que olvidar la fuerte componente matemática en la que se basa el trabajo. El tratamiento de resultados matemáticos conlleva una especial atención a la formalidad, y a expresar los conceptos e ideas de manera clara y precisa. Personalmente, como estudiante del Doble Grado en Ingeniería Informática y Matemáticas, poder combinar ambas disciplinas en este proyecto final ha resultado una experiencia muy satisfactoria.

Para concluir, querría poner en valor las técnicas que se han tratado, relacionadas con el FDA . Tanto el contraste ANOVA como los contrastes de medias tienen infinidad de aplicaciones a problemas reales, como se ha querido reflejar en este texto. El hecho de ponerlas a disposición del publico general supone un impacto social positivo, y más aún durante el curso en el que se ha elaborado el trabajo, bajo una crisis a escala global. Esta librería puede resultar de utilidad para muchos profesionales que se han volcado en encontrar soluciones a algunos problemas derivados de esta situación excepcional.

# Bibliografía


[1] M. Carbajo. Fda-py: desarrollo de un paquete python para el análisis de datos funcionales. *Universidad Autónoma de Madrid*, 2018.

[2] J.B. Conway. *A Course in Functional Analysis*. Graduate Texts in Mathematics. Springer New York, 1994. URL: https://books.google.es/books?id=ix4P1e6AkeIC.

[3] A. Cuevas, M. Febrero-Bande, and R. Fraiman. An anova test for functional data. *Computational Statistics & Data Analysis*, 47:111–122, 2004. doi:10.1016/j.csda.2003.10.021.

[4] G. van Rossum D. Goodger. *PEP 257 – Docstring Conventions*. Python, 2001. URL: https://www.python.org/dev/peps/pep-0257/.

[5] M. Febrero-Bande and M. Oviedo de la Fuente. Statistical computing in functional data analysis: The R package fda.usc. *Journal of Statistical Software*, 51(4):1–28, 2012. URL: http://www.jstatsoft.org/v51/i04/.

[6] F. Ferraty and P. Vieu. *Nonparametric Functional Data Analysis: Theory and Practice (Springer Series in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[7] R. A. Fisher. The arrengement of field experiments. *Journal of the Ministry of Agriculture of Great Britain*, 33:503–513, 1926. doi:https://dx.doi.org/10.23637\%2Frothamsted.8v20y.

[8] N. Coghlan G. van Rossum, B. Warsaw. *PEP 8 – Style Guide for Python Code*. Python, 2001. URL: https://www.python.org/dev/peps/pep-0008/.

[9] GAA-UAM. scikit-fda. https://https://github.com/GAA-UAM/scikit-fda, 2020. doi:10.5281/zenodo.3468127.

[10] L. Horváth and P. Kokoszka. *Inference for Functional Data with Applications*. Springer Series in Statistics. Springer New York, 2012. URL: https://books.google.es/books?id=OVezLB__ZpYC.

[11] T. Hsing and R. Eubank. *Theoretical Foundations of Functional Data Analysis, with an Introduction to Linear Operators*. Wiley Series in Probability and Statistics. Wiley, 2015. URL: https://books.google.es/books?id=h6MKCAAAQBAJ.

[12] P. Kokoszka and M. Reimherr. *Introduction to Functional Data Analysis*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press, 2017. URL: https://books.google.es/books?id=aHE3DwAAQBAJ.

[13] P. C. Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, 2:49–55, 1936.

[14] R.G.J. Miller. *Simultaneous Statistical Inference*. Springer Series in Statistics. Springer New York, 2012. URL: https://books.google.es/books?id=4C7VBwAAQBAJ.

[15] R. Olshen, E. Biden, M. Wyatt, and D. Sutherland. Gait analysis and the bootstrap. *Ann. Statist.*, 17(4):1419–1440, 12 1989. URL: https://doi.org/10.1214/aos/1176347372, doi:10.1214/aos/1176347372.

[16] A. Pini, A. Stamm, and S. Vantini. Hotelling's t2 in separable hilbert spaces. *Journal of Multivariate Analysis*, 167:284 – 305, 2018. URL: http://www.sciencedirect.com/science/article/pii/S0047259X17302646, doi:https://doi.org/10.1016/j.jmva.2018.05.007.

[17] J. O. Ramsay, H. Wickham, S. Graves, and G. Hooker. *fda: Functional Data Analysis*, 2018. R package version 2.4.8. URL: https://CRAN.R-project.org/package=fda.

[18] J.O. Ramsay. When the data are functions. *Psychometrika*, 47:379–396, 1982. doi:10.1007/BF02293704.

[19] J.O. Ramsay and B.W. Silverman. *Functional Data Analysis*. Springer series in statistics. Springer, 1997. URL: https://books.google.es/books?id=vYXCsgEACAAJ.

[20] M.R. Spiegel, J.J. Schiller, and R.A. Srinivasan. *Probabilidad y estadistica*. McGraw-Hill. Educación. McGraw-Hill Interamericana, 2013. URL: https://books.google.es/books?id=orpJygEACAAJ.

# Acrónimos

**AEMET** Agencia Estatal de Meteorología de España.

**ANOVA** Análisis de la varianza.

**API** Interfaz de programación de aplicaciones.

**FDA** Análisis de Datos Funcionales.

# Apéndices

# A

# Código de los ejemplos

[1]:
```
from skfda import datasets, concatenate
from skfda.datasets import make_gaussian_process
from skfda.misc.covariances import WhiteNoise
from skfda.representation import basis, FDataGrid

from skfda.inference.anova import oneway_anova, v_sample_stat, v_asymptotic_stat
from skfda.inference.hotelling import hotelling_test_ind
```

[2]:
```
import matplotlib
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.metrics import mean_squared_error
```

[3]:
```
matplotlib.rc('font', **{'size': 15})
sns.set_style('whitegrid')
```

[4]:
```
r_state = 42
```

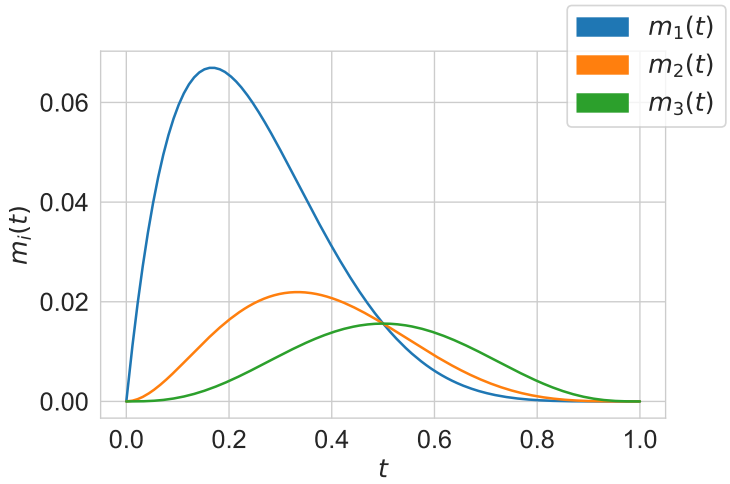
## 1 Estudio con datos sintéticos

[5]:
```
n_samples = 10
n_features = 100
n_groups = 3
start = 0
stop = 1

t = np.linspace(start, stop, n_features)

m2 = t ** 2 * (1 - t) ** 4
m3 = t ** 3 * (1 - t) ** 3
m1 = t * (1 - t) ** 5

FDataGrid([m1, m2, m3],
          dataset_label="",
          axes_labels=['$t$', '$m_i(t)$']
         ).plot(group=['$m_1(t)$', '$m_2(t)$', '$m_3(t)$'], legend=True)
plt.savefig('tfg_img/std_scatter.pdf');
```




[6]:
```
def add_noise(n_samples, m1, m2, m3, n_features, sigma2, start=0, stop=1):
    cov = WhiteNoise(variance=sigma2)
    fd1 = make_gaussian_process(n_samples, mean=m1, cov=cov, random_state=1,
                                n_features=n_features, start=start, stop=stop)
    fd2 = make_gaussian_process(n_samples, mean=m2, cov=cov, random_state=2,
                                n_features=n_features, start=start, stop=stop)
    fd3 = make_gaussian_process(n_samples, mean=m3, cov=cov, random_state=3,
                                n_features=n_features, start=start, stop=stop)
    return fd1, fd2, fd3
```

### 1.1 Nivel de ruido muy bajo ($\sigma^2 = 0.001$)

[7]:
```
sigma2 = 0.001
fd1, fd2, fd3 = add_noise(n_samples, m1, m2, m3, n_features, sigma2)
```

[8]:
```
%%time
stat, p_val = oneway_anova(fd1, fd2, fd3, n_reps=20000, random_state=r_state)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Statistic: 0.024
p-value: 0.000
CPU times: user 14.1 s, sys: 199 ms, total: 14.3 s
Wall time: 14.2 s
```

[9]:
```
stat, p_val = oneway_anova(fd1, fd2, n_reps=20000, random_state=r_state)
print('ANOVA test: fd1 vs fd2')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
ANOVA test: fd1 vs fd2
Statistic: 0.010
p-value: 0.000
```

[10]:
```
stat, p_val = hotelling_test_ind(fd1, fd2)
print('Hotelling test: fd1 vs fd2')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Hotelling test: fd1 vs fd2
Statistic: 18.749
p-value: 0.001
```

[11]:
```
stat, p_val = oneway_anova(fd1, fd3, n_reps=20000, random_state=r_state)
print('ANOVA test: fd1 vs fd3')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
ANOVA test: fd1 vs fd3
Statistic: 0.012
p-value: 0.000
```

[12]:
```
stat, p_val = hotelling_test_ind(fd1, fd3)
print('Hotelling test: fd1 vs fd3')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Hotelling test: fd1 vs fd3
Statistic: 15.402
p-value: 0.003
```

[13]:
```
stat, p_val = oneway_anova(fd2, fd3, n_reps=20000, random_state=r_state)
print('ANOVA test: fd2 vs fd3')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
ANOVA test: fd2 vs fd3
Statistic: 0.002
p-value: 0.296
```

[14]:
```
stat, p_val = hotelling_test_ind(fd2, fd3)
print('Hotelling test: fd2 vs fd3')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Hotelling test: fd2 vs fd3
Statistic: 4.678
p-value: 0.201
```

### 1.2 Nivel de ruido bajo ($\sigma^2 = 0.01$)

[15]:
```
sigma2 = 0.01
fd1, fd2, fd3 = add_noise(n_samples, m1, m2, m3, n_features, sigma2)
```

[18]:
```
stat, p_val = oneway_anova(fd1, fd2, fd3, n_reps=20000, random_state=r_state)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Statistic: 0.082
p-value: 0.070
```

[19]:
```
stat, p_val = oneway_anova(fd1, fd2, n_reps=20000, random_state=r_state)
print('ANOVA test: fd1 vs fd2')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
ANOVA test: fd1 vs fd2
Statistic: 0.030
p-value: 0.107
```

[20]:
```
stat, p_val = hotelling_test_ind(fd1, fd2)
print('Hotelling test: fd1 vs fd2')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Hotelling test: fd1 vs fd2
Statistic: 6.593
p-value: 0.166
```

[21]:
```
stat, p_val = oneway_anova(fd1, fd3, n_reps=20000, random_state=r_state)
print('ANOVA test: fd1 vs fd3')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
ANOVA test: fd1 vs fd3
Statistic: 0.032
p-value: 0.077
```

[22]:
```
stat, p_val = hotelling_test_ind(fd1, fd3)
print('Hotelling test: fd1 vs fd3')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Hotelling test: fd1 vs fd3
Statistic: 5.136
p-value: 0.342
```

[23]:
```
stat, p_val = oneway_anova(fd2, fd3, n_reps=20000, random_state=r_state)
print('ANOVA test: fd2 vs fd3')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
ANOVA test: fd2 vs fd3
Statistic: 0.021
p-value: 0.418
```

[24]:
```
stat, p_val = hotelling_test_ind(fd2, fd3)
print('Hotelling test: fd2 vs fd3')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Hotelling test: fd2 vs fd3
Statistic: 4.627
p-value: 0.218
```

### 1.3 Nivel de ruido medio ($\sigma^2 = 0.1$)

[25]:
```
sigma2 = 0.1
fd1, fd2, fd3 = add_noise(n_samples, m1, m2, m3, n_features, sigma2)
```

[26]:
```
stat, p_val = oneway_anova(fd1, fd2, fd3, n_reps=20000, random_state=r_state)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Statistic: 0.639
p-value: 0.358
```

[27]:
```
stat, p_val = oneway_anova(fd1, fd2, n_reps=20000, random_state=r_state)
print('ANOVA test: fd1 vs fd2')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
ANOVA test: fd1 vs fd2
Statistic: 0.202
p-value: 0.424
```

[28]:
```
stat, p_val = hotelling_test_ind(fd1, fd2)
print('Hotelling test: fd1 vs fd2')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Hotelling test: fd1 vs fd2
Statistic: 4.720
p-value: 0.427
```

[29]:
```
stat, p_val = oneway_anova(fd1, fd3, n_reps=20000, random_state=r_state)
print('ANOVA test: fd1 vs fd3')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
ANOVA test: fd1 vs fd3
Statistic: 0.227
p-value: 0.310
```

[30]:
```
stat, p_val = hotelling_test_ind(fd1, fd3)
print('Hotelling test: fd1 vs fd3')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Hotelling test: fd1 vs fd3
Statistic: 4.003
p-value: 0.590
```

[31]:
```
stat, p_val = oneway_anova(fd2, fd3, n_reps=20000, random_state=r_state)
print('ANOVA test: fd2 vs fd3')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
ANOVA test: fd2 vs fd3
Statistic: 0.209
p-value: 0.416
```

[32]:
```
stat, p_val = hotelling_test_ind(fd2, fd3)
print('Hotelling test: fd2 vs fd3')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Hotelling test: fd2 vs fd3
Statistic: 4.707
p-value: 0.205
```

### 1.4 Nivel de ruido alto ($\sigma^2 = 1$)

[33]:
```
sigma2 = 1
fd1, fd2, fd3 = add_noise(n_samples, m1, m2, m3, n_features, sigma2)
```

[34]:
```
stat, p_val = oneway_anova(fd1, fd2, fd3, n_reps=20000, random_state=r_state)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Statistic: 6.121
p-value: 0.433
```

[35]:
```
stat, p_val = oneway_anova(fd1, fd2, n_reps=20000, random_state=r_state)
print('ANOVA test: fd1 vs fd2')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
ANOVA test: fd1 vs fd2
Statistic: 1.847
p-value: 0.521
```

[36]:
```
stat, p_val = hotelling_test_ind(fd1, fd2)
print('Hotelling test: fd1 vs fd2')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Hotelling test: fd1 vs fd2
Statistic: 4.325
p-value: 0.517
```

[37]:
```
stat, p_val = oneway_anova(fd1, fd3, n_reps=20000, random_state=r_state)
print('ANOVA test: fd1 vs fd3')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
ANOVA test: fd1 vs fd3
Statistic: 2.171
p-value: 0.355
```

[39]:
```
stat, p_val = hotelling_test_ind(fd1, fd3)
print('Hotelling test: fd1 vs fd3')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Hotelling test: fd1 vs fd3
Statistic: 3.857
p-value: 0.629
```

[40]:
```
stat, p_val = oneway_anova(fd2, fd3, n_reps=20000, random_state=r_state)
print('ANOVA test: fd2 vs fd3')
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
ANOVA test: fd2 vs fd3
Statistic: 2.104
p-value: 0.412
```

[41]:
```
stat, p_val = hotelling_test_ind(fd2, fd3)
print('Hotelling test: fd2 vs fd3')
print("Statistic: {:.3f}".format(stat))
```

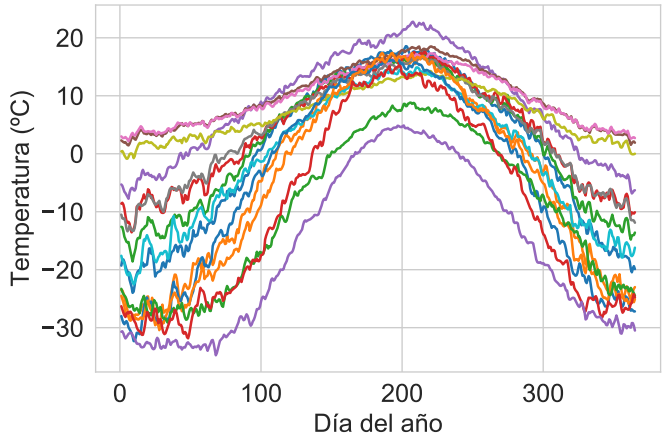
```
print("p-value: {:.3f}".format(p_val))
```

```
Hotelling test: fd2 vs fd3
Statistic: 4.742
p-value: 0.199
```

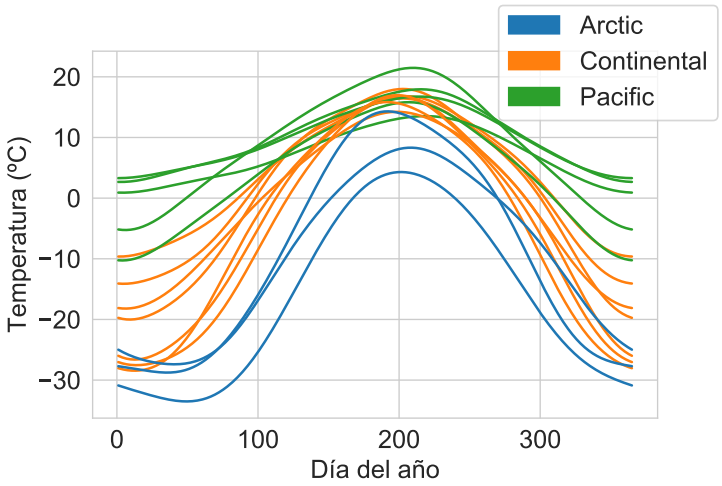
## 2 Estudio meteorológico de Canadá

[42]:
```
max_samples = 45
n_basis = 9
```

[43]:
```
weather = datasets.fetch_weather()
fd = weather['data'][20:max_samples].coordinates[0]
target = weather['target'][20:max_samples]
target_feature_names = weather['target_feature_names']
target_names = weather['target_names']
fd_basis = fd.to_basis(basis.Fourier(n_basis=n_basis))
fd.copy(dataset_label='', axes_labels=["Día del año", "Temperatura (ºC)"]).plot()
plt.savefig('tfg_img/canadian_1.pdf');
```




[44]:
```
labels = target_names[target]
fd_basis = fd_basis.copy(dataset_label='', axes_labels=["Día del año",␣
↪"Temperatura (ºC)"])
fd_basis.plot(group=labels, legend=True)
plt.savefig('tfg_img/canadian_basis.pdf');
```




[45]:
```
fd_arctic = fd_basis[labels == 'Arctic']
fd_continental = fd_basis[labels == 'Continental']
fd_pacific = fd_basis[labels == 'Pacific']
```

### 2.1 Contraste múltiple

[46]:
```
%%time
stat, p_val = oneway_anova(fd_arctic, fd_continental, fd_pacific, n_reps=2000,␣
 ↪random_state=r_state)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Statistic: 946770.388
p-value: 0.000
CPU times: user 2.57 s, sys: 105 ms, total: 2.68 s
Wall time: 1.9 s
```

### 2.2 Análisis post-hoc

[50]:
```
%%time
stat, p_val = hotelling_test_ind(fd_arctic, fd_pacific)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Statistic: 3710.080
p-value: 0.000
CPU times: user 72.6 ms, sys: 11.6 ms, total: 84.2 ms
Wall time: 129 ms
```

[51]:
```
%%time
stat, p_val = hotelling_test_ind(fd_arctic, fd_continental)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Statistic: 250.953
p-value: 0.025
CPU times: user 153 ms, sys: 4.5 ms, total: 158 ms
Wall time: 218 ms
```

[52]:
```
%%time
stat, p_val = hotelling_test_ind(fd_pacific, fd_continental)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Statistic: 182.386
p-value: 0.206
CPU times: user 600 ms, sys: 7.7 ms, total: 608 ms
Wall time: 661 ms
```

## 3 Estudio meteorológico de España

[53]:
```
aemet = datasets.fetch_aemet()
data = aemet['data'].coordinates[0]
meta = aemet['meta']
```

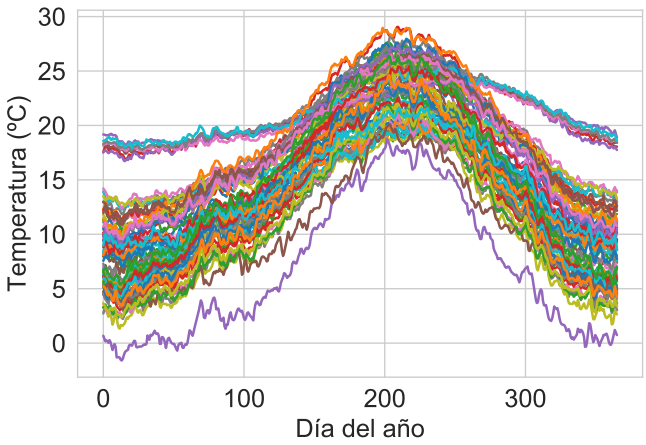
```
      ind                              name    province  altitude  year.ini  \
0    1387                          A CORUÑA    A CORUÑA        58      1980
1    1387                 A CORUÑA/ALVEDRO    A CORUÑA        98      1980
2    1428  SANTIAGO DE COMPOSTELA/LABACOLLA    A CORUÑA       370      1980
3   90910                   VITORIA/FORONDA       ALAVA       513      1980
4    8175              ALBACETE/LOS LLANOS    ALBACETE       704      1980
..    ...                              ...         ...       ...       ...
68   2539          VALLADOLID (VILLANUBLA)  VALLADOLID       846      1980
69   1082                BILBAO/AEROPUERTO     VIZCAYA        42      1980
70   2614                           ZAMORA      ZAMORA       656      1980
71   9390                           DAROCA    ZARAGOZA       779      1980
72   9434           ZARAGOZA (AEROPUERTO)    ZARAGOZA       247      1980

    year.end  longitude   latitude
0       2009  -8.419444  43.367222
1       2009  -8.372222  43.306944
2       2009  -8.410833  42.887778
3       2009  -2.733333  42.871944
4       2009  -1.863056  38.952222
..       ...        ...        ...
68      2009  -4.850000  41.700000
```

```
69      2009  -2.905833  43.298056
70      2009  -5.733611  41.516667
71      2009  -1.410833  41.114722
72      2009  -1.008056  41.661944

[73 rows x 8 columns]
```

[54]:
```
data.copy(dataset_label='', axes_labels=["Día del año", "Temperatura (ºC)"]).
 ↪plot()
plt.savefig('tfg_img/aemet_1.pdf');
```




[55]:
```
df = pd.read_csv('aemet.csv')
clima = df['clima']
clima[clima == 'MEDITERRANEO'] = 'Mediterráneo'
clima[clima == 'OCEANICO'] = 'Oceánico'
clima[clima == 'INTERIOR'] = 'Interior'
c = np.logical_or(np.logical_or(clima == 'Mediterráneo', clima == 'Oceánico'),␣
 ↪clima == 'Interior')

n_int, n_oce, n_med = 5, 7, 10
labels = np.full(n_int + n_oce + n_med, 'Mediterráneo')
labels[:n_int] = 'Interior'
labels[n_int:n_int + n_oce] = 'Oceánico'
```

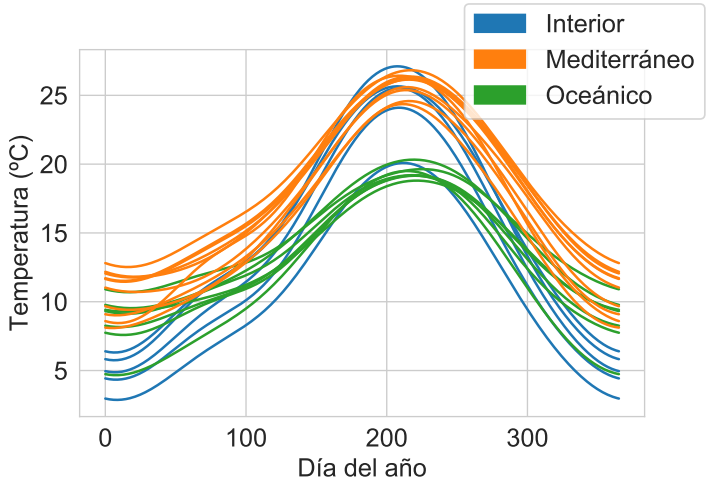
[56]:
```
meta_f = meta[c]
data_f = data[c]
fd_int = data_f[clima[c] == 'Interior'][:n_int]
fd_oce = data_f[clima[c] == 'Oceánico'][:n_oce]
fd_med = data_f[clima[c] == 'Mediterráneo'][:n_med]
```

### 3.1 Número de bases $k = 9$

[57]:
```
k = 9

fdb_int = fd_int.to_basis(basis.Fourier(n_basis=k))
fdb_oce = fd_oce.to_basis(basis.Fourier(n_basis=k))
fdb_med = fd_med.to_basis(basis.Fourier(n_basis=k))

concatenate([fdb_int, fdb_oce, fdb_med]).copy(dataset_label='',
                                              axes_labels=["Día del año",␣
 ↪"Temperatura (ºC)"]
                                             ).plot(group=labels, legend=True)
plt.savefig('tfg_img/aemet_basis9.pdf');
```




#### 3.1.1 Contraste múltiple

[59]:
```
stat, p_val = oneway_anova(fdb_int, fdb_oce, fdb_med, n_reps=20000,␣
 ↪random_state=r_state)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Statistic: 89709.344
p-value: 0.000
```

#### 3.1.2 Análisis post-hoc

[60]:
```
stat, p_val = hotelling_test_ind(fdb_med, fdb_int)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Statistic: 404.270
p-value: 0.005
```

[62]:
```
stat, p_val = hotelling_test_ind(fdb_med, fdb_oce)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Statistic: 2178.715
p-value: 0.000
```

[63]:
```
stat, p_val = hotelling_test_ind(fdb_int, fdb_oce)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

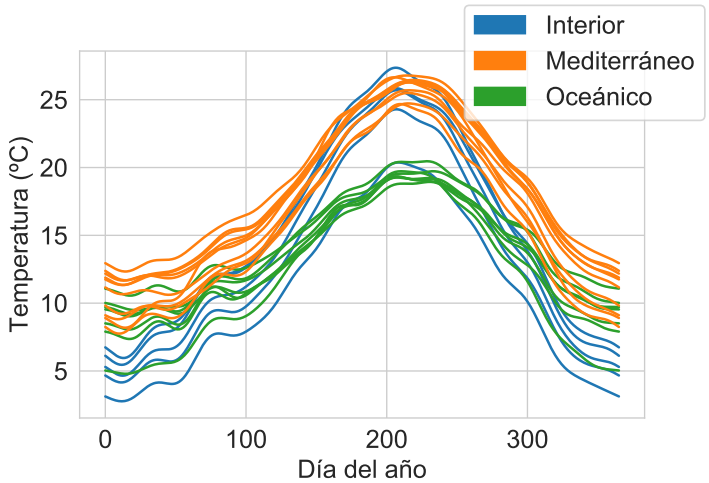
```
Statistic: 865.958
p-value: 0.039
```

### 3.2 Número de bases $k = 25$

[64]:
```
k = 25

fdb_int = fd_int.to_basis(basis.Fourier(n_basis=k))
fdb_oce = fd_oce.to_basis(basis.Fourier(n_basis=k))
fdb_med = fd_med.to_basis(basis.Fourier(n_basis=k))

concatenate([fdb_int, fdb_oce, fdb_med]).copy(dataset_label='',
                                              axes_labels=["Día del año",␣
 ↪"Temperatura (ºC)"]
                                             ).plot(group=labels, legend=True)
plt.savefig('tfg_img/aemet_basis25.pdf');
```



### 3.2.1 Contraste múltiple

[65]:
```
stat, p_val = oneway_anova(fdb_int, fdb_oce, fdb_med, n_reps=20000,␣
↪random_state=r_state)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Statistic: 89849.008
p-value: 0.000
```

### 3.2.2 Análisis post-hoc

[66]:
```
stat, p_val = hotelling_test_ind(fdb_med, fdb_int)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

```
Statistic: 447.998
p-value: 0.002
```

[67]:
```
stat, p_val = hotelling_test_ind(fdb_med, fdb_oce)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

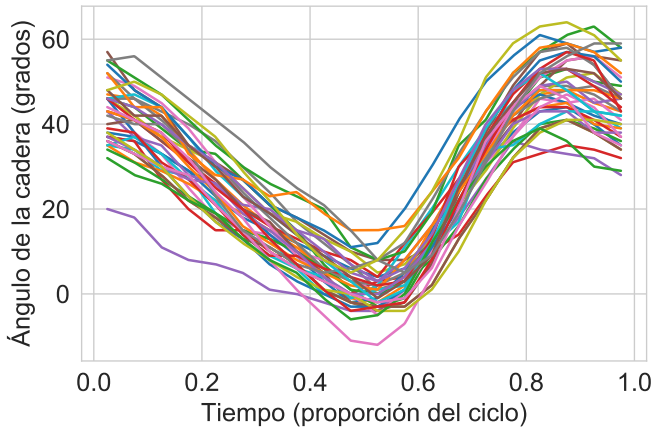
```
Statistic: 5200.611
p-value: 0.000
```

[68]:
```
stat, p_val = hotelling_test_ind(fdb_int, fdb_oce)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

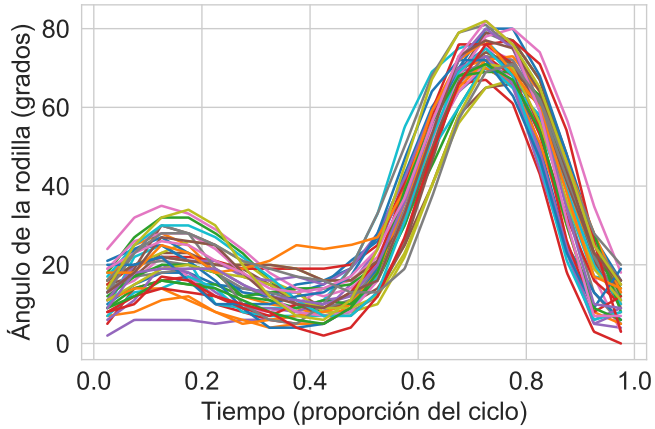
```
Statistic: 133.821
p-value: 0.003
```

# 4 Convergencia del p-valor

[69]:
```
fd = datasets.fetch_gait()['data']
fd_hip = fd.coordinates[0]
fd_knee = fd.coordinates[1]
fd_hip.copy(dataset_label='',
            axes_labels=["Tiempo (proporción del ciclo)", "Ángulo de la cadera␣
↪(grados)"]
           ).plot()
plt.gcf().subplots_adjust(bottom=0.15)
plt.savefig('tfg_img/gait_hip.pdf');
```




[70]:
```
fd_knee.copy(dataset_label='',
             axes_labels=["Tiempo (proporción del ciclo)", "Ángulo de la rodilla␣
 ↪(grados)"]
            ).plot()
plt.gcf().subplots_adjust(bottom=0.15)
plt.savefig('tfg_img/gait_knee.pdf');
```




[71]:
```
fd1 = fd_knee[:13]
fd2 = fd_knee[13:26]
fd3 = fd_knee[26:]
```

[72]:
```
n_samples = 50
min_exponent, max_exponent = 6, 17
```

```
indices = np.arange(min_exponent, max_exponent)
```

### 4.1 Convergencia del p-valor en ANOVA

[73]:
```
samples = np.empty((n_samples, max_exponent - min_exponent))
for i in range(n_samples):
    for j, exponent in enumerate(indices):
        print(i, j)
        samples[i][j] = oneway_anova(fd1, fd2, fd3,
                                     n_reps=2**exponent,
                                     equal_var=False)[1]
```

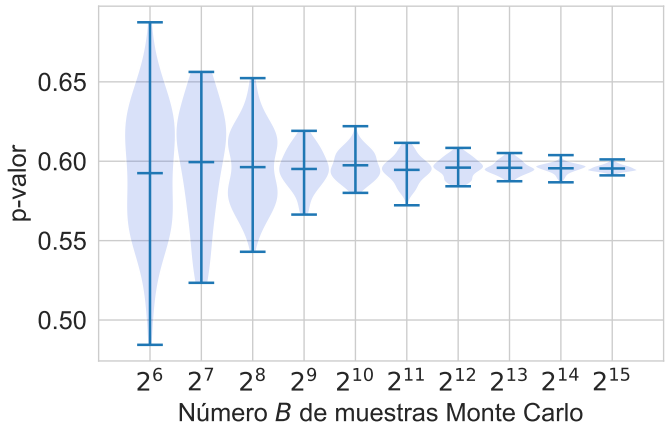
[74]:
```
samples_df = pd.DataFrame(columns=[str(i) for i in range(n_samples)],
                          index=np.arange(min_exponent, max_exponent))
for i, col in enumerate(samples_df.columns):
    samples_df[col] = datos[:, i]
samples_df.to_csv('p_val_anova.csv')
```

[75]:
```
samples_df = pd.read_csv('p_val_anova.csv', index_col=0)
samples = samples_df.to_numpy()
x = 2 ** samples_df.index
```

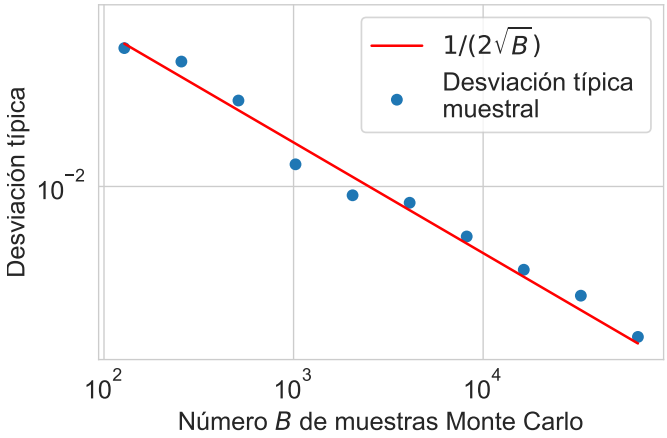
[76]:
```
pos = np.linspace(1, 5.5, 10) # Escala para el eje horizontal
label = ["$2^{%d}$" % i for i in indices]

plt.figure()
ax = plt.subplot(111)
parts = plt.violinplot(list(samples), pos, showmeans=True)
for pc in parts['bodies']:
    pc.set_facecolor('royalblue')

    pc.set_alpha(0.2)
ax.set_xticks(pos)
ax.set_xticklabels(label)
plt.xlabel("Número $B$ de muestras Monte Carlo")
plt.ylabel("p-valor")
plt.gcf().subplots_adjust(bottom=0.15)
plt.savefig('tfg_img/violin_anova.pdf');
```




[77]:
```
plt.scatter(x, np.std(samples, axis=1), label='Desviación típica \nmuestral')
plt.plot(x, 1/(2 * np.sqrt(x)), "r", label=r'$1/(2\sqrt{B})$')
plt.xscale("log")
plt.yscale("log")
plt.xlabel("Número $B$ de muestras Monte Carlo\n")
plt.ylabel("Desviación típica")
plt.legend()
plt.gcf().subplots_adjust(bottom=0.15)
plt.savefig('tfg_img/scatter_anova.pdf', pad_inches=2);
```




[78]:
```
mean_squared_error(1/(2 * np.sqrt(x)), np.std(samples, axis=1))
```

[78]:
```
5.279112035228594e-06
```

## 4.2 Convergencia del p-valor en $T^2$ de Hotelling

[79]:
```
samples = np.empty((n_samples, max_exponent - min_exponent))
for i in range(n_samples):
    for j, exponent in enumerate(indices):
        print(i, j)
        samples[i][j] = hotelling_test_ind(fd1, fd2, n_reps=2**exponent)[1]
```

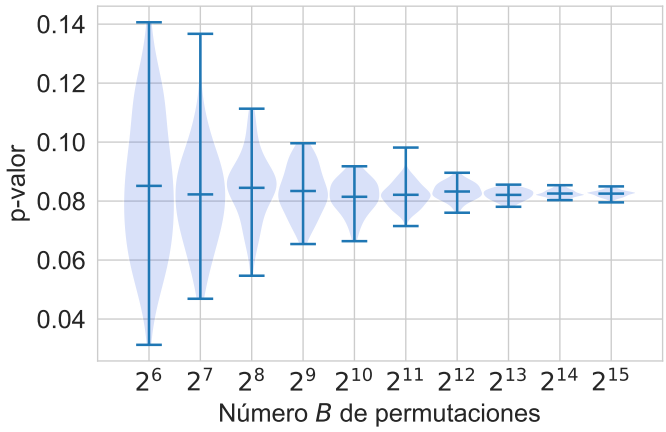
[80]:
```
samples_df = pd.DataFrame(columns=[str(i) for i in range(n_samples)],
                          index=np.arange(min_exponent, max_exponent))
for i, col in enumerate(samples_df.columns):
    samples_df[col] = datos[:, i]
samples_df.to_csv('p_val_t2.csv')
```

[81]:
```
samples_df = pd.read_csv('p_val_t2.csv', index_col=0)
samples = samples_df.to_numpy()
x = 2 ** samples_df.index
```

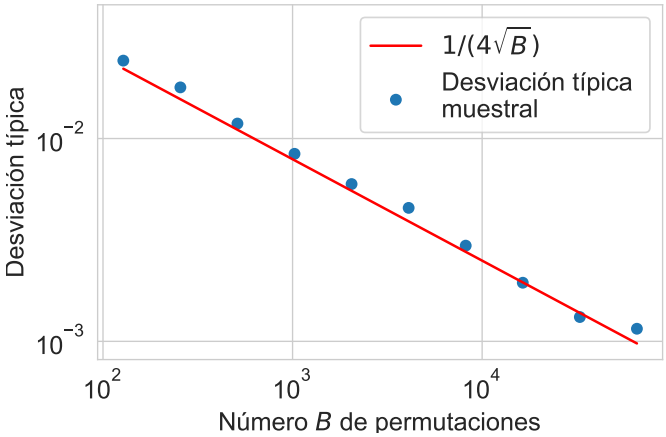
[82]:
```
pos = np.linspace(1, 5.5, 10) # Escala para el eje horizontal
label = ["$2^{%d}$" % i for i in indices]

plt.figure()
ax = plt.subplot(111)
parts = plt.violinplot(list(samples), pos, showmeans=True)
for pc in parts['bodies']:
    pc.set_facecolor('royalblue')

    pc.set_alpha(0.2)
ax.set_xticks(pos)
ax.set_xticklabels(label)
plt.xlabel("Número $B$ de permutaciones")
plt.ylabel("p-valor")
plt.gcf().subplots_adjust(bottom=0.15)
plt.savefig('tfg_img/violin_t2.pdf')
```




[83]:
```
plt.scatter(x, np.std(samples, axis=1), label='Desviación típica \nmuestral')
plt.plot(x, 1/(4 * np.sqrt(x)), "r", label=r'$1/(4\sqrt{B})$')
plt.xscale("log")
plt.yscale("log")
plt.xlabel("Número $B$ de permutaciones\n")
plt.ylabel("Desviación típica")
plt.legend()
plt.gcf().subplots_adjust(bottom=0.15)
plt.savefig('tfg_img/scatter_t2.pdf');
```




[84]:
```
mean_squared_error(1/(2 * np.sqrt(x)), np.std(samples, axis=1))
```

[84]: 7.845754076121961e-05

# B

# Manual del programador

Docs » API Reference » Inference » ANOVA

# ANOVA

This package groups a collection of statistical models, useful for analyzing equality of means for different subsets of a sample.

## One-way functional ANOVA

Functionality to perform One-way ANOVA analysis, to compare means among different samples. One-way stands for one functional response variable and one unique variable of input.

| | |
|---|---|
| `skfda.inference.anova.oneway_anova`(*args[, ...]) | Performs one-way functional ANOVA. |

## Statistics

Statistics that measure the internal and external variability between groups, used in the models above.

| | |
|---|---|
| `skfda.inference.anova.v_sample_stat`(fd, weights) | Calculates a statistic that measures t |
| `skfda.inference.anova.v_asymptotic_stat`(fd, ...) | Calculates a statistic that measures t |

# oneway_anova

**skfda.inference.anova.oneway_anova**(**args, n_reps=2000, return_dist=False, random_state=None, p=2*) [source]

Performs one-way functional ANOVA.

This function implements an asymptotic method to test the following null hypothesis:

Let $\{X_i\}_{i=1}^k$ be a set of $k$ independent samples each one with $n_i$ trajectories, and let $E(X_i) = m_i(t)$. The null hypothesis is defined as:

$$H_0 : m_1(t) = \cdots = m_k(t)$$

This function calculates the value of the statistic `v_sample_stat()` $V_n$ with the means of the given samples. Under the null hypothesis this statistic is asymptotically equivalent to `v_asymptotic_stat()`, where each sample is replaced by a gaussian process, with mean zero and the same covariance function as the original.

The simulation of the distribution of the asymptotic statistic $V$ is implemented using a bootstrap procedure. One observation of the $k$ different gaussian processes defined above is simulated, and the value of `v_asymptotic_stat()` is calculated. This procedure is repeated *n_reps* times, creating a sampling distribution of the statistic.

This procedure is from Cuevas[1].

**Parameters**

- **fd1,fd2,...** (*FDataGrid*) – The sample measurements for each each group.
- **n_reps** (*int*, *optional*) – Number of simulations for the bootstrap procedure. Defaults to 2000 (This value may change in future versions).
- **return_dist** (*bool*, *optional*) – Flag to indicate if the function should return a numpy.array with the sampling distribution simulated.
- **random_state** (*optional*) – Random state.
- **p** (*int*, *optional*) – p of the lp norm. Must be greater or equal than 1. If p='inf' or p=np.inf it is used the L infinity metric. Defaults to 2.

**Returns**

Value of the sample statistic, p-value and sampling distribution of the simulated asymptotic statistic.

**Return type:**

(float, float, numpy.array)

**Raises**

ValueError – In case of bad arguments.

### Examples

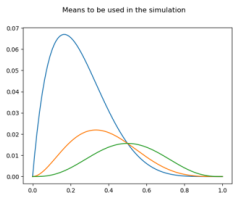
```
>>> from skfda.inference.anova import oneway_anova
>>> from skfda.datasets import fetch_gait
>>> from numpy.random import RandomState
>>> from numpy import printoptions
```

```
>>> fd = fetch_gait()["data"].coordinates[1]
>>> fd1, fd2, fd3 = fd[:13], fd[13:26], fd[26:]
>>> oneway_anova(fd1, fd2, fd3, random_state=RandomState(42))
(179.52499999999998, 0.602)
>>> _, _, dist = oneway_anova(fd1, fd2, fd3, n_reps=3,
...     random_state=RandomState(42),
...     return_dist=True)
>>> with printoptions(precision=4):
...     print(dist)
[ 163.3577 208.595  229.7678]
```

### References

[1] Antonio Cuevas, Manuel Febrero-Bande, and Ricardo Fraiman. “An anova test for functional data”. *Computational Statistics Data Analysis*, 47:111-112, 02 2004

## Examples using `skfda.inference.anova.oneway_anova`




*One-way functional ANOVA with synthetic data*

# v_sample_stat

**skfda.inference.anova.v_sample_stat**(*fd, weights, p=2*) [source]

Calculates a statistic that measures the variability between groups of samples in a `skfda.representation.FData` object.

The statistic defined as below is calculated between all the samples in a `skfda.representation.FData` object with a given set of weights.

Let $\{f_i\}_{i=1}^k$ be a set of samples in a FData object. Let $\{w_j\}_{j=1}^k$ be a set of weights, where $w_i$ is related to the sample $f_i$ for $i = 1, \dots, k$. The statistic is defined as:

$$V_n = \sum_{i<j}^{k} w_i \|f_i - f_j\|^2$$

This statistic is defined in Cuevas[1].

**Parameters**

- **fd** (*FData*) – Object containing all the samples for which we want to calculate the statistic.
- **weights** – Weights related to each sample. Each weight is expected to appear in the same position as its corresponding sample in the FData object.

**Returns**

The value of the statistic.

**Raises**

ValueError –

**Examples**

```
>>> from skfda.inference.anova import v_sample_stat
>>> from skfda.representation.grid import FDataGrid
>>> import numpy as np
```

We create different trajectories to be applied in the statistic and a set of weights.

```
>>> t = np.linspace(0, 1, 50)
>>> x1 = t * (1 - t) ** 5
>>> x2 = t ** 2 * (1 - t) ** 4
>>> x3 = t ** 3 * (1 - t) ** 3
>>> fd = FDataGrid([x1, x2, x3], sample_points=t)
>>> weights = [10, 20, 30]
```

Finally the value of the statistic is calculated:

```
>>> v_sample_stat(fd, weights)
0.01649448843348894
```

### References

[1] Antonio Cuevas, Manuel Febrero-Bande, and Ricardo Fraiman. “An anova test for functional data”. *Computational Statistics Data Analysis*, 47:111-112, 02 2004

/

# v_asymptotic_stat

**skfda.inference.anova.v_asymptotic_stat**(*fd, weights, p=2*) [source]

Calculates a statistic that measures the variability between groups of samples in a `skfda.representation.FData` object.

The statistic defined as below is calculated between all the samples in a `skfda.representation.FData` object with a given set of weights.

Let $\{f_i\}_{i=1}^k$ be a set of samples in a FData object. Let $\{w_j\}_{j=1}^k$ be a set of weights, where $w_i$ is related to the sample $f_i$ for $i = 1, \dots, k$. The statistic is defined as:

$$\sum_{i<j}^{k} \|f_i - f_j \sqrt{\frac{w_i}{w_j}}\|^2$$

This statistic is defined in Cuevas[1].

**Parameters**

- **fd** (*FData*) – Object containing all the samples for which we want to calculate the statistic.
- **weights** – Weights related to each sample. Each weight is expected to appear in the same position as its corresponding sample in the FData object.

**Returns**

The value of the statistic.

**Raises**

ValueError –

### Examples

```
>>> from skfda.inference.anova import v_asymptotic_stat
>>> from skfda.representation.grid import FDataGrid
>>> import numpy as np
```

We create different trajectories to be applied in the statistic and a set of weights.

```
>>> t = np.linspace(0, 1, 50)
>>> x1 = t * (1 - t) ** 5
>>> x2 = t ** 2 * (1 - t) ** 4
>>> x3 = t ** 3 * (1 - t) ** 3
>>> fd = FDataGrid([x1, x2, x3], sample_points=t)
>>> weights = [10, 20, 30]
```

Finally the value of the statistic is calculated:

```
>>> v_asymptotic_stat(fd, weights)
0.0018159320335885969
```

### References

[1] Antonio Cuevas, Manuel Febrero-Bande, and Ricardo Fraiman. “An anova test for functional data”. *Computational Statistics Data Analysis*, 47:111-112, 02 2004

/

# Hotelling

This package groups a collection of statistical tests based on Hotelling's statistic.

| | |
|---|---|
| `skfda.inference.hotelling.hotelling_t2`(fd1, fd2) | Calculates Hotelling's $T^2$ over two s |
| `skfda.inference.hotelling.hotelling_test_ind`(...) | Calculate the $T^2$-test for the means |

# hotelling_t2

**skfda.inference.hotelling.hotelling_t2**(*fd1*, *fd2*, *weights=None*) [source]

Calculates Hotelling's $T^2$ over two samples in `skfda.representation.FData` objects with sizes $n_1$ and $n_2$ as defined below:

$$T^2 = n(\mathbf{m}_1 - \mathbf{m}_2)^\top \mathbf{W}^{1/2}(\mathbf{W}^{1/2}\mathbf{K}_{\text{pooled}}\mathbf{W}^{1/2})^+\mathbf{W}^{1/2}(\mathbf{m}_1 - \mathbf{m}_2),$$

where $(\cdot)^+$ indicates the Moore-Penrose pseudo-inverse operator, $n = n_1 + n_2$, $W$ is a matrix of weights (usually Gram matrix), and $\mathbf{m}_1$, $\mathbf{m}_2$ are the means of each ample, $\mathbf{K}_{\text{pooled}}$ matrix is defined as

$$\mathbf{K}_{\text{pooled}} := \frac{n_1 - 1}{n_1 + n_2 - 2}\mathbf{K}_{n_1} + \frac{n_2 - 1}{n_1 + n_2 - 2}\mathbf{K}_{n_2},$$

where $\mathbf{K}_{n_1}$, $\mathbf{K}_{n_2}$ are the sample covariance matrices, computed with the basis coefficients or using the discrete representation, depending on the input.

This statistic is defined in Pini, Stamm and Vantini[1].

**Parameters**

- **fd1** (*FData*) – Object with the first sample.
- **fd2** (*FData*) – Object containing second sample.
- **weights** (*numpy.array*, *optional*) – Weights matrix. If no value is passed then uses Gram matrix if data is in basis representation. Identity matrix is used for discretized data.

**Returns**

The value of the statistic.

**Raises**

**TypeError.** –

### Examples

```
>>> from skfda.inference.hotelling import hotelling_t2
>>> from skfda.representation import FDataGrid, basis
```

```
>>> fd1 = FDataGrid([[1, 1, 1], [3, 3, 3]])
>>> fd2 = FDataGrid([[3, 3, 3], [5, 5, 5]])
>>> '%.2f' % hotelling_t2(fd1, fd2)
'2.00'
>>> fd1 = fd1.to_basis(basis.Fourier(n_basis=3))
>>> fd2 = fd2.to_basis(basis.Fourier(n_basis=3))
>>> '%.2f' % hotelling_t2(fd1, fd2)
'2.00'
```

### References

[1] A. Pini, A. Stamm and S. Vantini, “Hotelling’s t2 in separable hilbert spaces”, *Jounal of Multivariate Analysis*, 167 (2018), pp.284-305.

/

Docs » API Reference » Inference » Hotelling » hotelling_test_ind

# hotelling_test_ind

**skfda.inference.hotelling.hotelling_test_ind**(*fd1, fd2, n_reps=None, random_state=None, return_dist=False*) [source]

Calculate the $T^2$-test for the means of two independent samples of functional data.

This is a two-sided test for the null hypothesis that 2 independent samples have identical average (expected) values. This test assumes that the populations have identical variances by default.

The p-value of the test is calculated using a permutation test over the statistic `hotelling_t2()`. If a maximum number of repetitions of the algorithm is provided then the permutations tested are generated randomly.

This procedure is from Pini, Stamm and Vantinni[1].

**Parameters**

- **fd1,fd2** (*FData*) – Samples of data. The FData objects must have the same type.
- **n_reps** (*int*, *optional*) – Maximum number of repetitions to compute p-value. Default value is None.
- **random_state** (*optional*) – Random state.
- **return_dist** (*bool*, *optional*) – Flag to indicate if the function should return a numpy.array with the values of the statistic computed over each permutation.

**Returns**

Value of the sample statistic, one tailed p-value and a collection of statistic values from permutations of the sample.

**Return type:**

(float, float, numpy.array)

**Raises**

TypeError – In case of bad arguments.

**Examples**

```
>>> from skfda.inference.hotelling import hotelling_t2
>>> from skfda.representation import FDataGrid, basis
>>> from numpy import printoptions
```

```
>>> fd1 = FDataGrid([[1, 1, 1], [3, 3, 3]])
>>> fd2 = FDataGrid([[3, 3, 3], [5, 5, 5]])
>>> t2n, pval, dist = hotelling_test_ind(fd1, fd2, return_dist=True)
>>> '%.2f' % t2n
'2.00'
>>> '%.2f' % pval
'0.00'
>>> with printoptions(precision=4):
...     print(dist)
[ 2. 2. 0. 0. 2. 2.]
```

### References

[1] A. Pini, A. Stamm and S. Vantini, “Hotelling’s t2 in separable hilbert spaces”, *Jounal of Multivariate Analysis*, 167 (2018), pp.284-305.

/

Docs » Examples » One-way functional ANOVA with synthetic data

**Note**

Click here to download the full example code

# One-way functional ANOVA with synthetic data

This example shows how to perform a functional one-way ANOVA test with synthetic data.

```
# Author: David García Fernández
# License: MIT


import numpy as np

from skfda.representation import FDataGrid
from skfda.inference.anova import oneway_anova
from skfda.datasets import make_gaussian_process
from skfda.misc.covariances import WhiteNoise
```

*One-way ANOVA* (analysis of variance) is a test that can be used to compare the means of different samples of data. Let $X_{ij}(t), j = 1, \dots, n_i$ be trajectories corresponding to $k$ independent samples ($i = 1, \dots, k$) and let $E(X_i(t)) = m_i(t)$. Thus, the null hypothesis in the statistical test is:

$$H_0 : m_1(t) = \cdots = m_k(t)$$

In this example we will explain the nature of ANOVA method and its behavior under certain conditions simulating data. Specifically, we will generate three different trajectories, for each one we will simulate a stochastic process by adding to them white noise. The main objective of the test is to illustrate the differences in the results of the ANOVA method when the covariance function of the brownian processes changes.

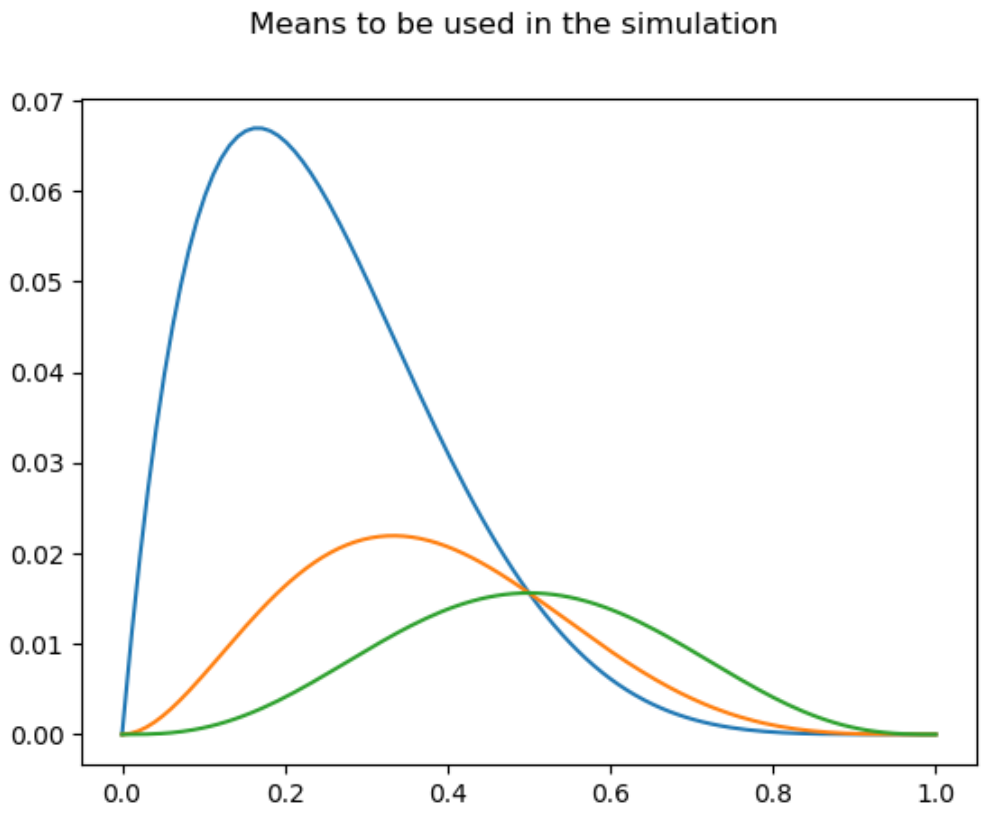
First, the means for the future processes are drawn.

```
n_samples = 10
n_features = 100
n_groups = 3
start = 0
stop = 1

t = np.linspace(start, stop, n_features)

m1 = t * (1 - t) ** 5
m2 = t ** 2 * (1 - t) ** 4
m3 = t ** 3 * (1 - t) ** 3

_ = FDataGrid([m1, m2, m3],
              dataset_label="Means to be used in the simulation").plot()
```




A total of *n_samples* trajectories will be created for each mean, so a array of labels is created to identify them when plotting.

```
groups = np.full(n_samples * n_groups, 'Sample 1')
groups[10:20] = 'Sample 2'
groups[20:] = 'Sample 3'
```

First simulation uses a low $\sigma^2 = 0.01$ value. In this case the differences between the means of each group should be clear, and the p-value for the test should be near to zero.

```
sigma2 = 0.01
cov = WhiteNoise(variance=sigma2)

fd1 = make_gaussian_process(n_samples, mean=m1, cov=cov,
                            n_features=n_features, random_state=1, start=start,
                            stop=stop)
fd2 = make_gaussian_process(n_samples, mean=m2, cov=cov,
                            n_features=n_features, random_state=2, start=start,
                            stop=stop)
fd3 = make_gaussian_process(n_samples, mean=m3, cov=cov,
                            n_features=n_features, random_state=3, start=start,
                            stop=stop)
stat, p_val = oneway_anova(fd1, fd2, fd3, random_state=4)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

Out:

```
Statistic: 0.082
p-value: 0.108
```

In the following, the same process will be followed incrementing sigma value, this way the differences between the averages of each group will be lower and the p-values will increase (the null hypothesis will be harder to refuse).

Plot for $\sigma^2 = 0.1$:

```
sigma2 = 0.1
cov = WhiteNoise(variance=sigma2)

fd1 = make_gaussian_process(n_samples, mean=m1, cov=cov,
                            n_features=n_features, random_state=1, start=t[0],
                            stop=t[-1])
fd2 = make_gaussian_process(n_samples, mean=m2, cov=cov,
                            n_features=n_features, random_state=2, start=t[0],
                            stop=t[-1])
fd3 = make_gaussian_process(n_samples, mean=m3, cov=cov,
                            n_features=n_features, random_state=3, start=t[0],
                            stop=t[-1])

stat, p_val = oneway_anova(fd1, fd2, fd3, random_state=4)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

Out:

```
Statistic: 0.639
p-value: 0.373
```

Plot for $\sigma^2 = 1$:

```
sigma2 = 1
cov = WhiteNoise(variance=sigma2)

fd1 = make_gaussian_process(n_samples, mean=m1, cov=cov,
                            n_features=n_features, random_state=1, start=t[0],
                            stop=t[-1])
fd2 = make_gaussian_process(n_samples, mean=m2, cov=cov,
                            n_features=n_features, random_state=2, start=t[0],
                            stop=t[-1])
fd3 = make_gaussian_process(n_samples, mean=m3, cov=cov,
                            n_features=n_features, random_state=3, start=t[0],
                            stop=t[-1])

stat, p_val = oneway_anova(fd1, fd2, fd3, random_state=4)
print("Statistic: {:.3f}".format(stat))
print("p-value: {:.3f}".format(p_val))
```

Out:

```
Statistic: 6.121
p-value: 0.421
```

**References:**


[1] Antonio Cuevas, Manuel Febrero-Bande, and Ricardo Fraiman. “An anova test for functional data”. *Computational Statistics Data Analysis*, 47:111-112, 02 2004


**Total running time of the script:** ( 0 minutes 4.196 seconds)

Download Python source code: plot_oneway_synthetic.py

Download Jupyter notebook: plot_oneway_synthetic.ipynb

Docs » Examples » One-way functional ANOVA with real data

**Note**

Click here to download the full example code

# One-way functional ANOVA with real data

This example shows how to perform a functional one-way ANOVA test using a real dataset.

```
# Author: David García Fernández
# License: MIT

# sphinx_gallery_thumbnail_number = 4

import skfda
from skfda.inference.anova import oneway_anova
from skfda.representation import FDataGrid, FDataBasis
from skfda.representation.basis import Fourier
```

*One-way ANOVA* (analysis of variance) is a test that can be used to compare the means of different samples of data. Let $X_{ij}(t), j = 1, \dots, n_i$ be trajectories corresponding to $k$ independent samples $(i = 1, \dots, k)$ and let $E(X_i(t)) = m_i(t)$. Thus, the null hypothesis in the statistical test is:

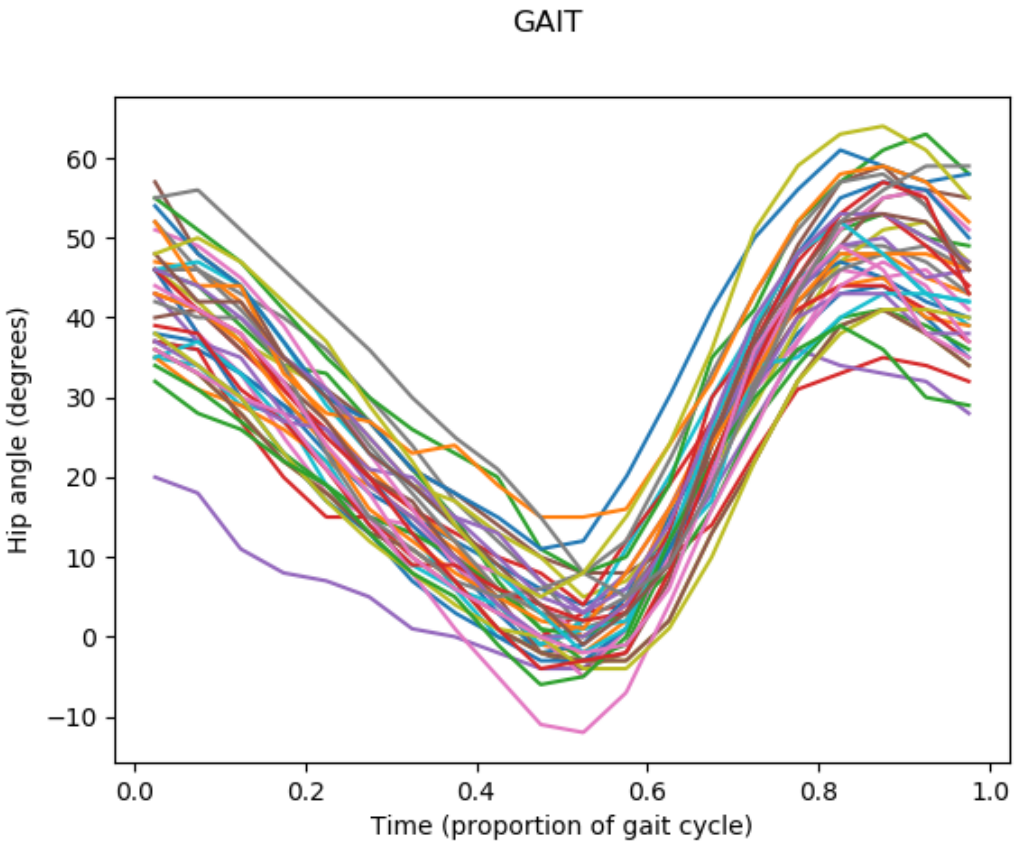
$$H_0 : m_1(t) = \cdots = m_k(t)$$

To illustrate this functionality we are going to explore the data available in GAIT dataset from *fda* R library. This dataset compiles a set of angles of hips and knees from 39 different boys in a 20 point movement cycle.

```
dataset = skfda.datasets.fetch_gait()
fd_hip = dataset['data'].coordinates[0]
fd_knee = dataset['data'].coordinates[1].to_basis(Fourier(n_basis=10))
```

Let's start with the first feature, the angle of the hip. The sample consists in 39 different trajectories, each representing the movement of the hip of each of the boys studied.

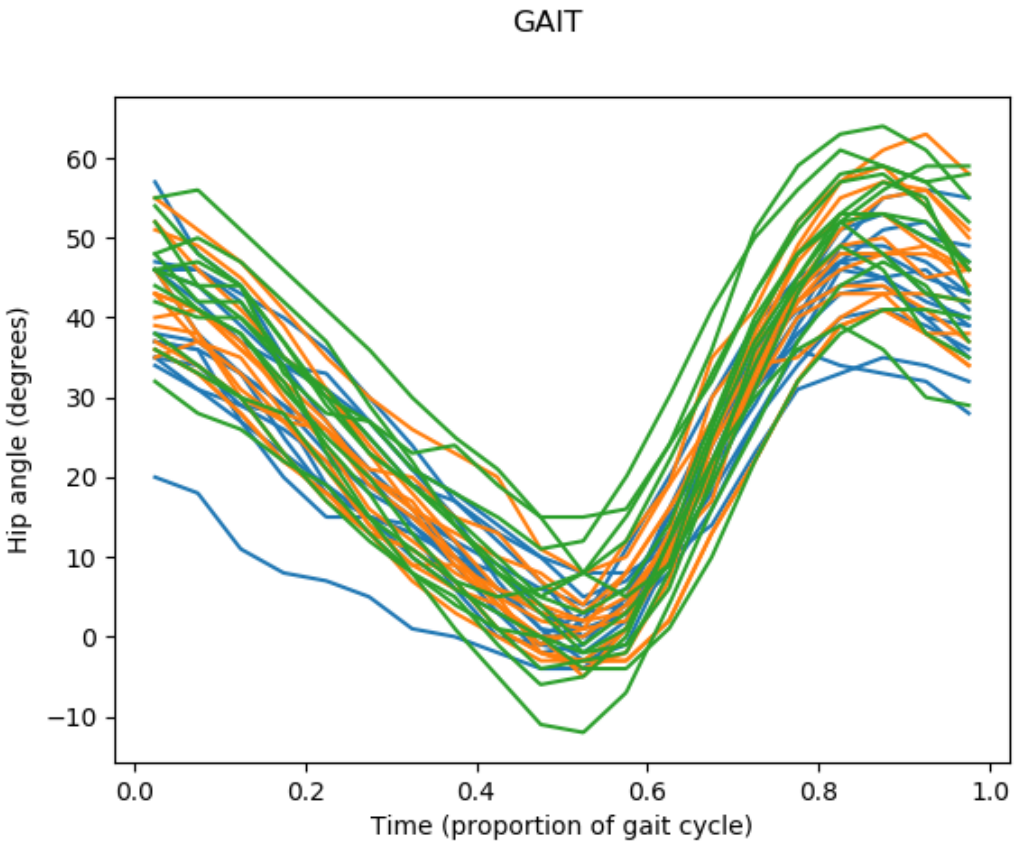
```
fig = fd_hip.plot()
```

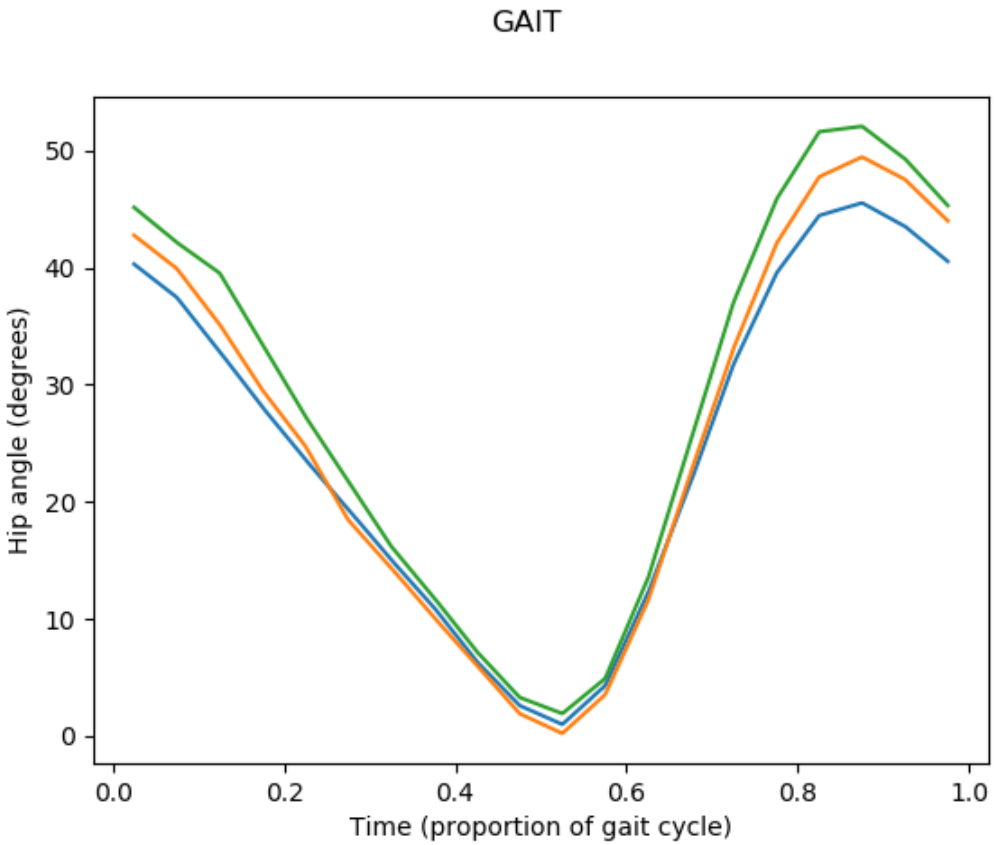



The example is going to be divided in three different groups. Then we are going to apply the ANOVA procedure to this groups to test if the means of this three groups are equal or not.

```
fd_hip1 = fd_hip[0:13]
fd_hip2 = fd_hip[13:26]
fd_hip3 = fd_hip[26:39]
fd_hip.plot(group=[0 if i < 13 else 1 if i < 26 else 39 for i in range(39)])

means = [fd_hip1.mean(), fd_hip2.mean(), fd_hip3.mean()]
fd_means = skfda.concatenate(means)
fig = fd_means.plot()
```




At this point is time to perform the *ANOVA* test. This functionality is implemented in the function `oneway_anova()`. As it consists in an asymptotic method it is possible to set the number of simulations necessary to approximate the result of the statistic. It is possible to set the $p$ of the $L_p$ norm used in the calculations (defaults 2).

```
v_n, p_val = oneway_anova(fd_hip1, fd_hip2, fd_hip3)
```

The function returns first the statistic `v_sample_stat()` used to measure the variability between groups, second the *p-value* of the test . For further information visit `oneway_anova()` and [1].

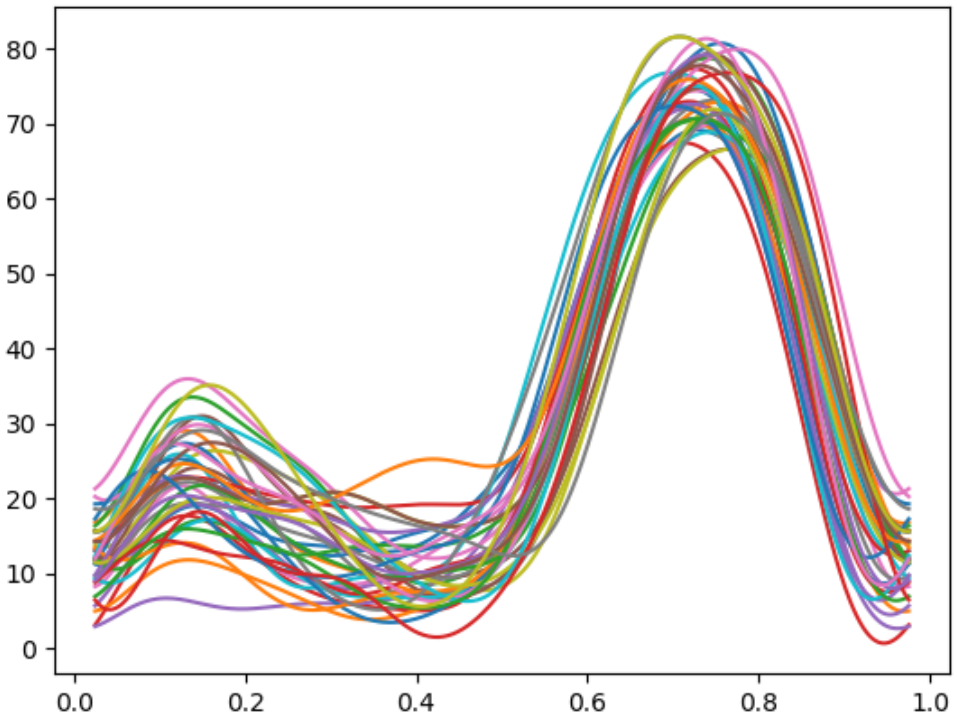
```
print('Statistic: ', v_n)
print('p-value: ', p_val)
```

Out:

```
Statistic:  373.02435897435896
p-value:  0.2
```

This was the simplest way to call this function. Let's see another example, this time using knee angles, this time with data in basis representation.

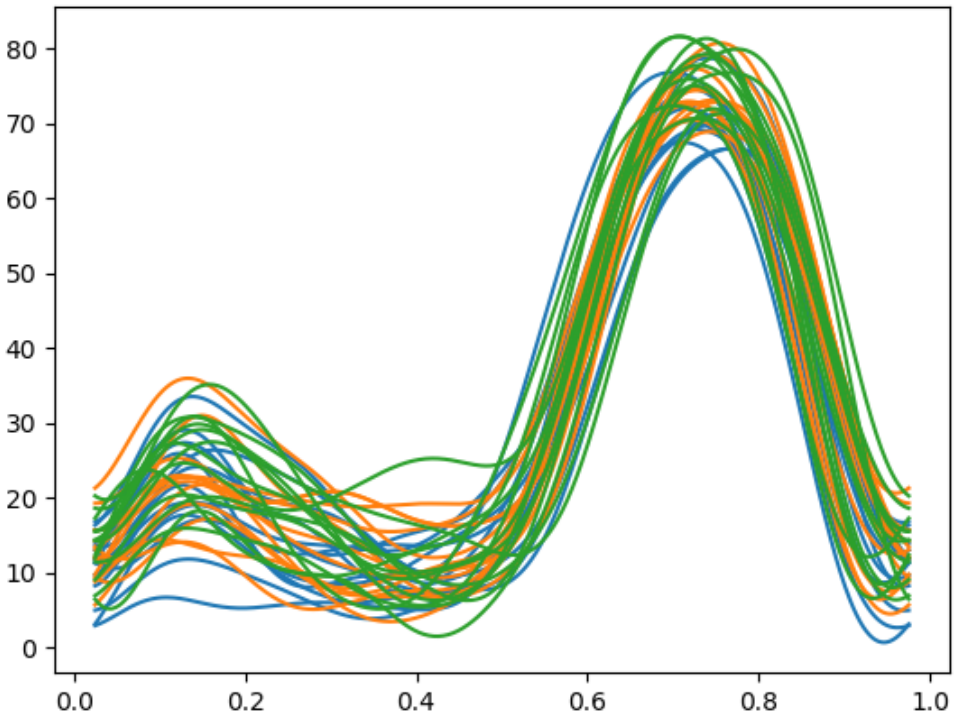
```
fig = fd_knee.plot()
```

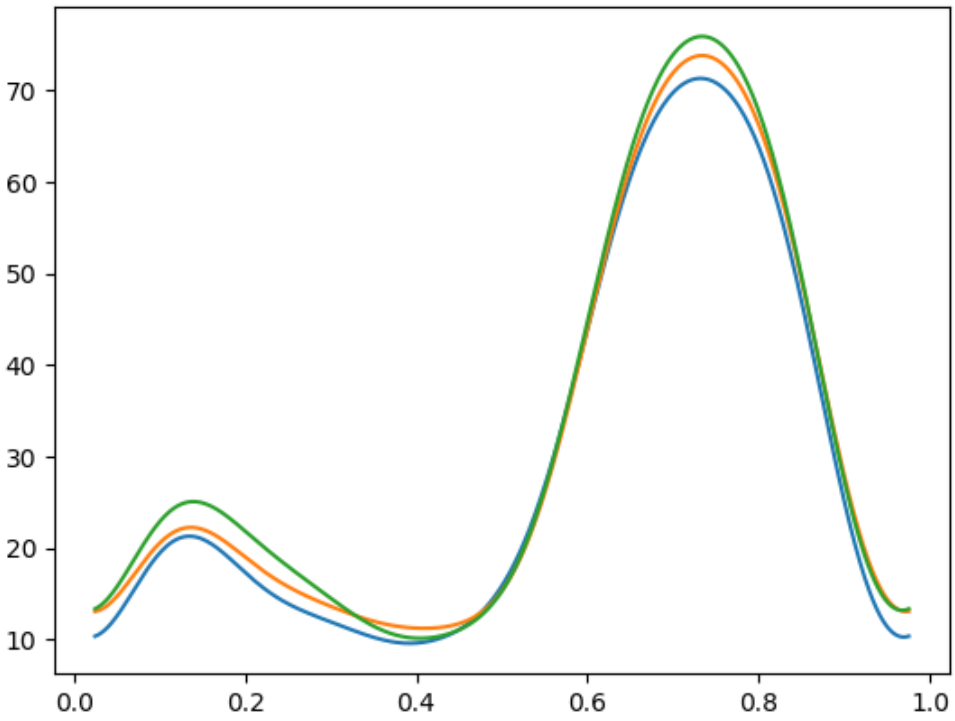


The same procedure as before is followed to prepare the data.

```
fd_knee1 = fd_knee[0:13]
fd_knee2 = fd_knee[13:26]
fd_knee3 = fd_knee[26:39]
fd_knee.plot(group=[0 if i < 13 else 1 if i < 26 else 39 for i in range(39)])

means = [fd_knee1.mean(), fd_knee2.mean(), fd_knee3.mean()]
fd_means = skfda.concatenate(means)
fig = fd_means.plot()
```




In this case the optional arguments of the function are going to be set. First, there is a *n_reps* parameter, which allows the user to select the number of simulations to perform in the asymptotic procedure of the test ( see `oneway_anova()` ), defaults to 2000.

Also there is a *p* parameter to choose the $p$ of the $L_p$ norm used in the calculations (defaults 2).

Finally we can set to True the flag *dist* which allows the function to return a third value. This third return value corresponds to the sampling distribution of the statistic which is compared with the first return to get the *p-value*.

```
v_n, p_val, dist = oneway_anova(fd_knee1, fd_knee2, fd_knee3, n_reps=1500,
                                return_dist=True)

print('Statistic: ', v_n)
print('p-value: ', p_val)
print('Distribution: ', dist)
```

Out:

```
Statistic:  178.58462912748067
p-value:  0.5866666666666667
Distribution:  [324.93202572 172.7557737  529.04108208 ... 139.62762772 445.23829203
 137.41883697]
```

**References:**

[1] Antonio Cuevas, Manuel Febrero-Bande, and Ricardo Fraiman. “An anova test for functional data”. *Computational Statistics Data Analysis*, 47:111-112, 02 2004

**Total running time of the script:** ( 0 minutes 3.394 seconds)

Download Python source code: plot_oneway.py

Download Jupyter notebook: plot_oneway.ipynb

Gallery generated by Sphinx-Gallery

/